W3School 菜鸟笔记 Android 基础入门教程

wizardforcel

Published with GitBook



目錄

W3School 菜鸟笔记 Android 基础入门教程

作者：coder-pig

来源：Android 基础入门教程

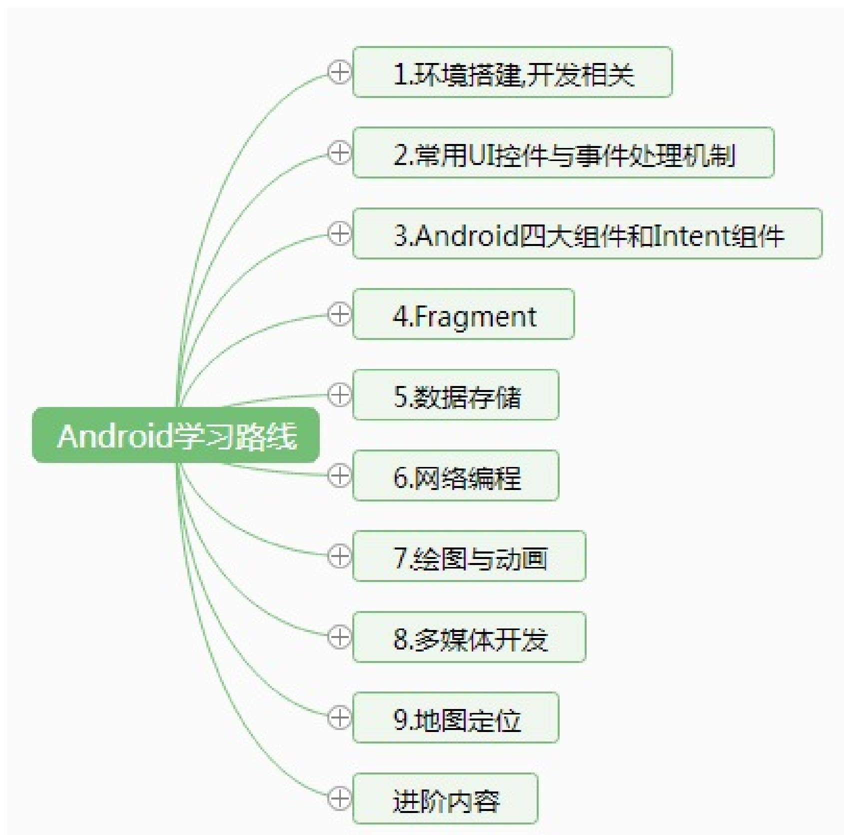
1.0 Android基础入门教程

一些唠嗑：

> 笔者是csdn的一名小博主，一名应届毕业生，你可以叫我coder-pig，你也可以和其他朋友一样称呼我为"小猪"，在csdn写blog差不多有一年多了，获益良多，写过几个入门的系列，C，Java以及Android，不过这些文章都是以前写的，工作一段时间后回头看自己的文章，感觉以前因为见识以及刚刚接触编程没多久，很多东西都写得浅薄，有很多局限，不足，最近一个月想了很久，觉得还是有必要重新来写一个完整的入门教程的，网上教程不同于书籍，书写了就不会变了，很多知识都是旧的，另外，天朝的书很多都是你抄我，我抄你的；小猪想让自己的教程结合实际开发，而并非是一堆专业名词，小猪一直都很重基础，因为面试的时候别人不会问你怎么去用一个第三方的控件，网络请求框架等等，而会问你基本的原理，很多人基本功不扎实，往往答不出个所以然，一些基本的东西还是要懂得！小猪的教程尽量以路线图 + 核心知识讲解 + 代码实例，来帮助大家学习Android！当然有些东西你一开始可能并不了解，不过你以后回过头来看，会发现这些东西很有用！好了，唠嗑就到这里！

教程路线图简介：

> 有图有真相，好的，小猪构思几天的教程路线图草图如下，后面可能会根据实际情况进行调整！ 脑图草图完整版可见： 百度脑图Android入门教程.km



> 先看下第一节的大概内容吧：



> 后续的内容会慢慢地放出来的！

```
/**
*
*----------Dragon be here!----------/
*    ┏┓   ┏┓
*   ┏┛┻━━━┛┻┓
*   ┃       ┃
*   ┃   ━   ┃
*   ┃ ┳┛  ┗┳ ┃
*   ┃       ┃
*   ┃   ┻   ┃
*   ┃       ┃
*   ┗━┓   ┏━┛
*     ┃   ┃神兽保佑
*     ┃   ┃代码无BUG！
*     ┃   ┗━━━┓
*     ┃       ┣┓
*     ┃       ┏┛
*     ┗┓┓┏━┳┓┏┛
*      ┃┫┫ ┃┫┫
*      ┗┻┛ ┗┻┛
* ━━━━━━神兽出没━━━━━━by:coder-pig
*/
```

1.0.1 2015年最新Android基础入门教程目录

前言：

关于《2015年最新Android基础入门教程目录》终于在今天落下了帷幕，全套教程共148节已编写完毕，附上目录，关于教程的由来，笔者的情况和自学心得，资源分享 以及一些疑问等可戳：《2015最新Android基础入门教程》完结散花~ 下面是本系列教程的完整目录：

后记：

> 以上就是关于《2015年安卓基础入门教程》的全部内容~ 假如本系列的教程为你学习Android开发带来一定的便利，不妨小额打赏下小猪，
>
> 
>
>
> 当然，不打赏也没什么，帮忙点个赞，留下你的评论，就是对小猪的支持了， 好了，就说这么多，谢谢~有问题请到小猪群反馈：小猪Android开发交流群

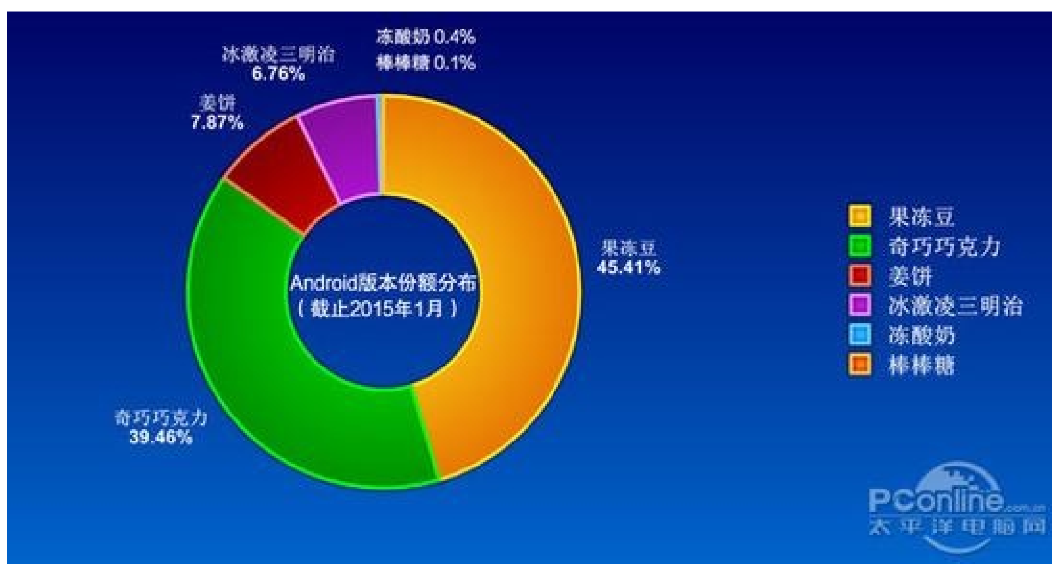
1.1 背景相关与系统架构分析

1.Android背景与当前的状况

Android系统是由Andy Rubin创建的，后来被Google收购了；最早的版本是:Android 1.1版本 而现在最新的版本是今年5.28，Google I/O大会上推出的Android M，有趣的是Android系统的命名都是以点心来命名的，下述表是15个Android版本名称，对应API号以及发布时间！

系统版本名称	API版本号	发布时间
Android 1.5：Cupcake：纸杯蛋糕	3	2009.4.30
Android 1.6：Donut：甜甜圈	4	2009.9.15
Android 2.0/2.0.1/2.1：Eclair：松饼	5/6/7	2009.10.26
Android 2.2/2.2.1：Froyo：冻酸奶	8	2010.5.20
Android 2.3：Gingerbread：姜饼	9	2010.12.7
Android 3.0：Honeycomb：蜂巢	11	2011.2.2
Android 3.1：Honeycomb：蜂巢	12	2011.5.11
Android 3.2：Honeycomb：蜂巢	13	2011.7.13
Android 4.0：Ice Cream Sandwich：冰激凌三文治	14	2011.10.19
Android 4.1：Jelly Bean：果冻豆	16	2012.6.28
Android 4.2：Jelly Bean：果冻豆	17	2012.10.30
Android 4.3：Jelly Bean：果冻豆	18	2013.7.25
Android 4.4：KitKat：奇巧巧克力	19	2013.11.01
Android 5.0：Lollipop：棒棒糖	21	2014.10.16
Android M：预览版	22	2015.5.28

好了，除了上面这些公共版本外，当然还有一些其他的版本，截止2015.1，各个版本的市场份额如下：



看完上面的信息，我们可能有这样的一个疑问： 那么多的系统版本，我们开发的时候要针对哪个版本进行开发？ 这就是作为一个Android必须面对的Android的"碎片化"问题了，而这个问题又分为两个： ①系统碎片化：我们开发App时可能需要做到低版本兼容，比如，最低兼容至2.3版本；由于各种Rom定制的盛行，国人都喜欢对原生系统做一些更改，这导致了在原生系统上可行，而在定制Rom上不可行的问题，比如相机调用~ ②屏幕碎片化：市面上各种各样屏幕尺寸的手机，4.3寸，4.5寸，4.7寸，5.0寸，5.3寸...等等，除了手机外，还有Android平板，所以开发时我们可能要处理这个屏幕适配的问题，当然，刚学我们并不需要去考虑这些复杂的东西，后续实际开发我们再来深究！

2.Android系统特性与平台架构

系统特性：

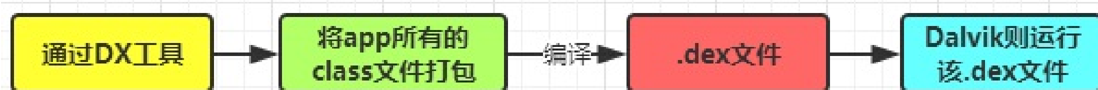
> - 应用程序框架支持组件的重用与替换（app发布时遵守了框架的约定，其他app也可以使用该模块）
> - **Dalvik**虚拟机:专门为移动设备优化 -集成的浏览器:开源的**WebKit**引擎
> - **SQLite**结构化的数据存储
> - 优化的图形库,多媒体支持,GSM电话技术,蓝牙等
> - 采用软件叠层方式构建

平台架构图：



架构的简单理解：

1. **Application(应用程序层)** 我们一般说的应用层的开发就是在这个层次上进行的，当然包括了系统内置的一组应用程序，使用的是Java语言
2. **Application Framework(应用程序框架层)** 无论系统内置或者我们自己编写的App，都需要使用到这层，比如我们想弄来电黑名单，自动挂断电话，我们就需要用到电话管理(TelephonyManager) 通过该层我们就可以很轻松的实现挂断操作，而不需要关心底层实现
3. **Libraries(库) + Android Runtime(Android运行时)** Android给我们提供了一组C/C++库，为平台的不同组件所使用，比如媒体框架；而Android Runtime则由Android核心库集 + Dalvik虚拟机构成，Dalvik虚拟机是针对移动设备的虚拟机，它的特点:不需要很快的CPU计算速度和大量的内存空间;而每个App都单独地运行在单独的Dalvik虚拟机内每个app对于一条Dalvik进程）而他的简单运行流程如：



4. **Linux内核** 这里就是涉及底层驱动的东西了，一些系统服务，比如安全性，内存管理以及进程管理等

3.本节小结：

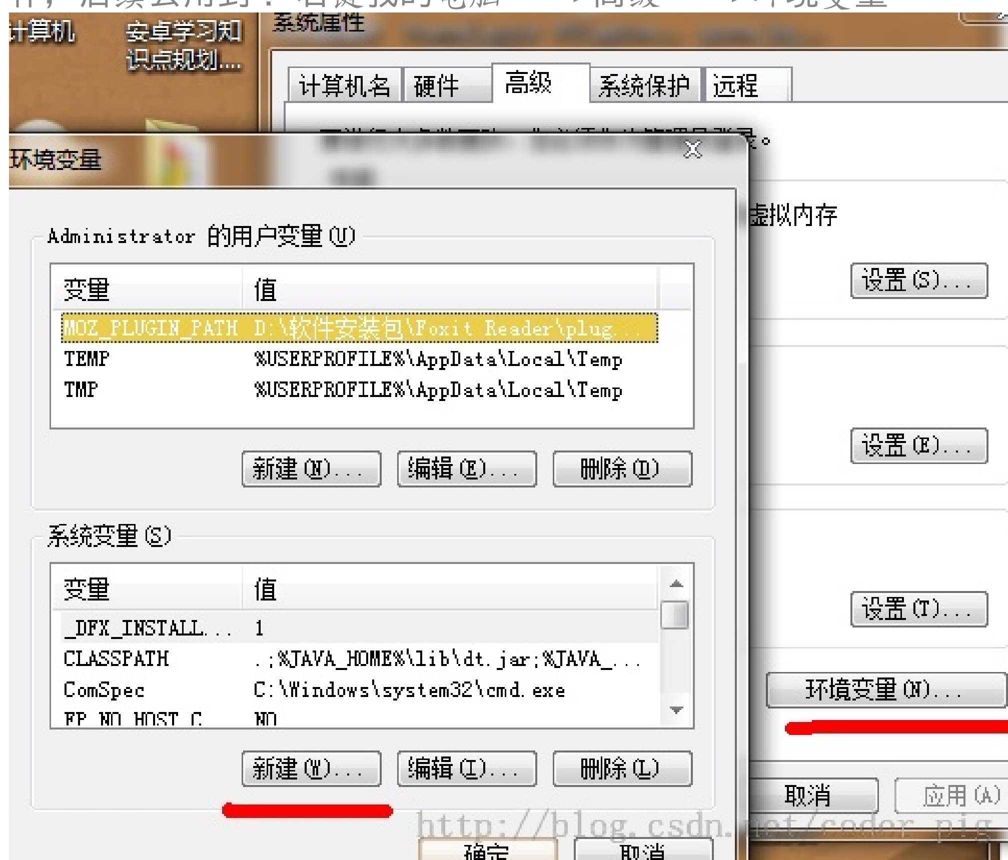
本节对Android的历史背景以及现状进行了了解，然后简单分析了Android的系统特性以及系统架构，这些概念性的东西，我们了解了解即可，而下一节我们将开始Android环境的搭建！

1.2 开发环境搭建

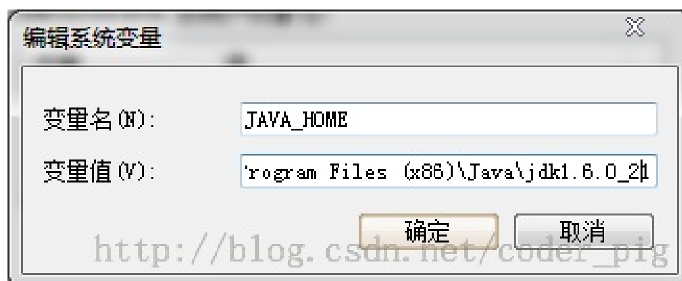
> 现在主流的Android开发环境有: ①Eclipse + ADT + SDK ②Android Studio + SDK ③IntelliJ IDEA + SDK 现在国内大部分开发人员还是使用的Eclipse，而谷歌宣布不再更新ADT后，并且官网也去掉了集成Android开发环境的Eclipse下载链接，各种现象都表示开发者最后都终将过渡到Android Studio，当然这段过渡时间会很长，但如果你是刚学Android的话建议直接冲Android Studio着手；而且很多优秀的开源项目都是基于Android Studio！当然，在本教程中对两种开发环境都会进行一个介绍，用哪个取决与你自己~ 还有一个IntelliJ，和Android Studio差不多的，并不对此进行讲解！

1.JDK安装与配置

- **Step 1:**下载**JDK** 可以到官网进行下载：Jdk官方下载 也可到笔者网盘下载：笔者网盘 PS:这个随便下一个都可以，关系不大，32位的只能下32位哦！
- **Step 2:JDK**安装 傻瓜式的下一步即可！
- **Step 3:**环境变量的配置 配置环境变量是为了方便我们一些命令行的操作，后续会用到！ 右键我的电脑——>高级——>环境变量



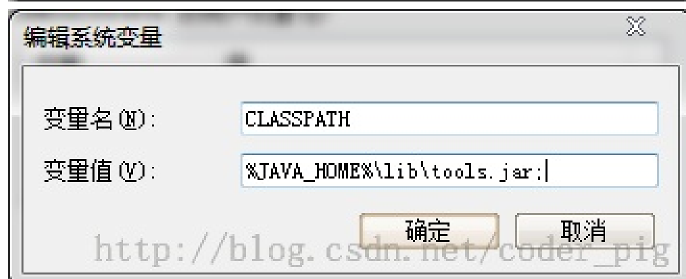
新建



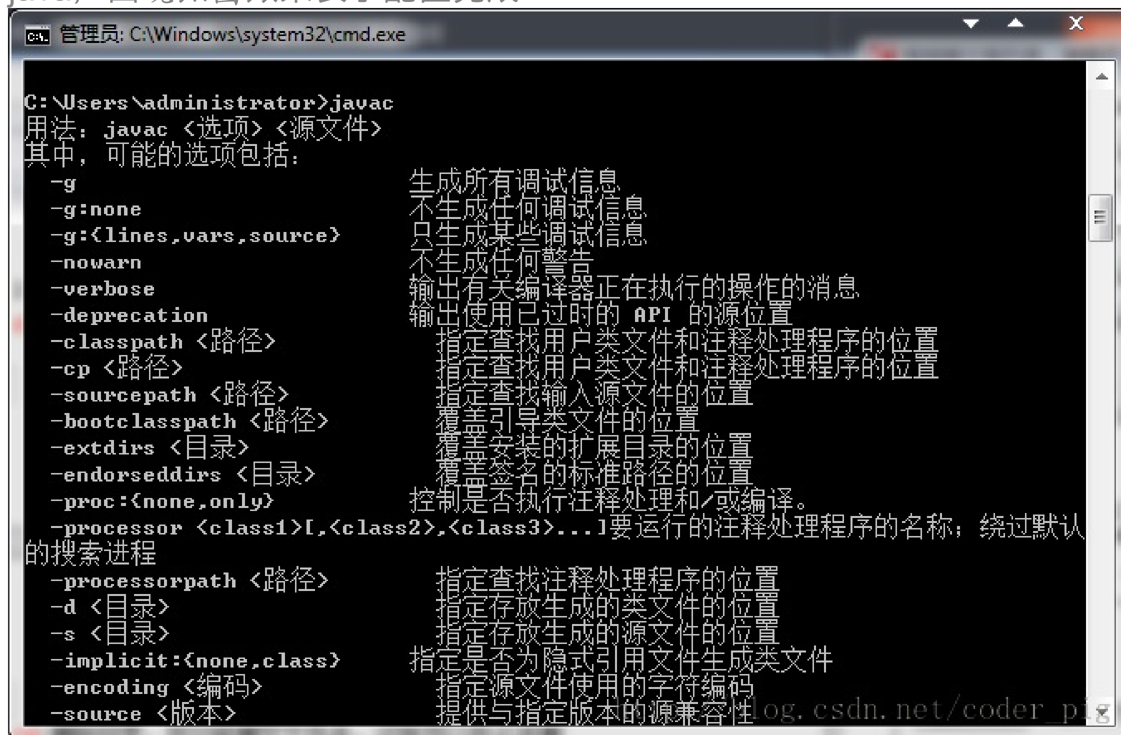
JAVA_HOME 修改**PATH**变量，别把原本的东西删掉！！！！



新建**CLASSPATH**



验证环境是否配置完成 打开电脑的cmd(命令行)，win键 + R输入cmd，然后在命令行依次javac和java，出现如图效果表示配置完成：



2.开发工具二选一

> 一开始也说了开发环境IDE的现状，另外前面忘记说一点:Android Studio是比较吃配置的，如果电脑不怎么好，建议还是先使用Eclipse进行Android开发，下面先说下我们熟悉IDE开发APP的流程，按照下述流程来熟悉IDE的使用：
>
> 1.如何创建一个工程
> 2.工程的目录结构的了解
> 3.与开发相关的常用视图
> 4.我们在哪里写代码
> 5.如何创建启动AVD安卓虚拟机
> 6.如何将写好的安卓程序加载到AVD上
> 7.当程序报错时,使用logcat查看日志信息
> 8.通过logcat信息,找到错误代码并改正
> 9.程序运行无误后签名打包成apk文件
>
>
>
> 然后下面两个选一个，开始我们的Android开发之路吧！
>
> **Eclipse + ADT + SDK**： **Android Studio + SDK**：

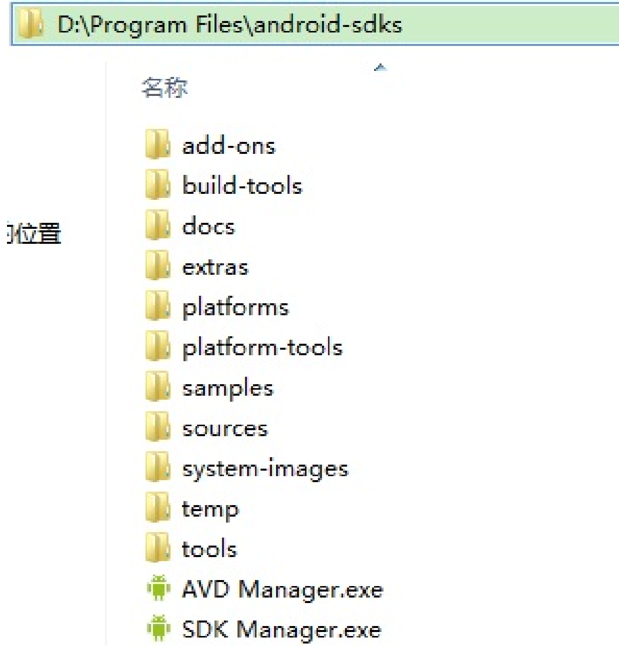
3.相关术语的解析

> 1. **Dalvik**： Android特有的虚拟机,和JVM不同,Dalvik虚拟机非常适合在移动终端上使用!
> 2. **AVD**： (android virtual machine):安卓虚拟设备,就是安卓的模拟器
> 3. **ADT**： (android development tools)安卓开发工具
> 4. **SDK**：(software development kit)软件开发工具包,就是安卓系统,平台架构等的工具集合,如adb.exe
> 5. **DDMS**：(dalvik debug monitor service)安卓调试工具
> 6. **adb**：安卓调试桥,在sdk的platform-tools目录下,功能很多,命令行必备
> 7. **DX**工具：将.class转换成.dex文件
> 8. **AAPT**：(android asset packing tool),安卓资源打包工具
> 9. **R.java**文件：由aapt工具根据App中的资源文件自动生成,可以理解为资源字典
> 10. **AndroidManifest.xml**：app包名 + 组件声明 + 程序兼容的最低版本 + 所需权限等程序的配置文件

后续内容对于初学者的你可能有点难度，但后面回头，你会发现这些东西很有用~看不懂的话，可以先跳过

4.ADB命令行的一些指令

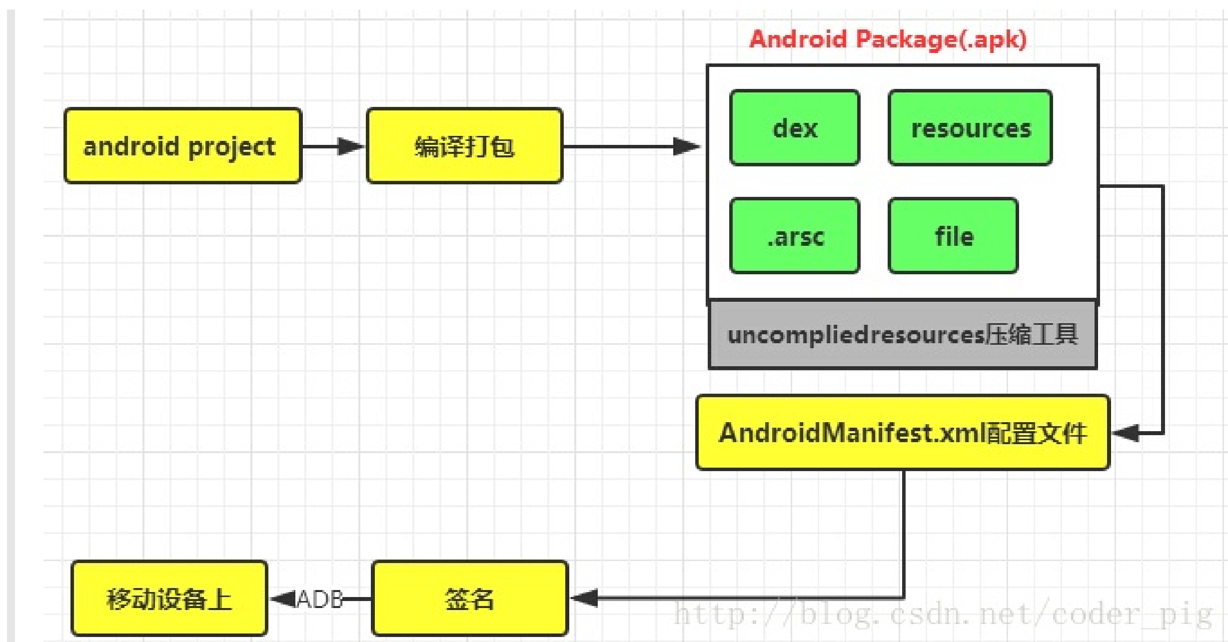
执行ADB指令之前我们还需要为我们的SDK配置一下环境变量 **Step 1:**新建一个ANDROID_HOME的环境变量，把sdk根目录地址贴上去：



Step 2:更新Path环境变量，在Paht开头加上：**%ANDROID_HOME%\tools;** 即可 配置完了，接下来就来学习指令了：



5.APP程序打包与安装的流程：



6.APP的安装过程：



7.本节小结

> 本节我们对Android开发IDE的现状进行了分析，建议初学者硬件条件允许的话，使用Android Studio来进行Android APP的开发，讲述了JDK的安装与配置，以及熟悉IDE的流程，一些关键名称的解析，ADB命令行的常用指令，最后还有程序的打包安装和安装过程的解析！经过这章相信大家对Android开发有了个简单了解，了解开发一个程序的流程，以及对应的项目目录结构！相信大家会抱怨模拟器AVD跑得很慢，下节会给大家介绍一个比真机还快的Android模拟器——Genymotion的安装使用！

1.2.1 使用Eclipse + ADT + SDK开发Android APP

1.前言

这里我们有两条路可以选，直接使用封装好的用于开发Android的ADT Bundle，或者自己进行配置 因为谷歌已经放弃了ADT的更新，官网上也取消的下载链接，这里提供谷歌放弃更新前最新版本的 ADT Bundle供大家下载！

2.直接使用打包好的Eclipse

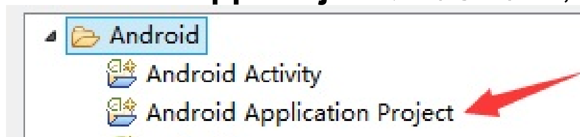
32位版：adt-bundle-windows-x86-20140702.zip **64**位版：adt-bundle-windows-x86_64-20140702.zip 下载解压，然后直接跳到4.来创建一个Helloworld工程！

3.自己配置Eclipse + ADT + SDK

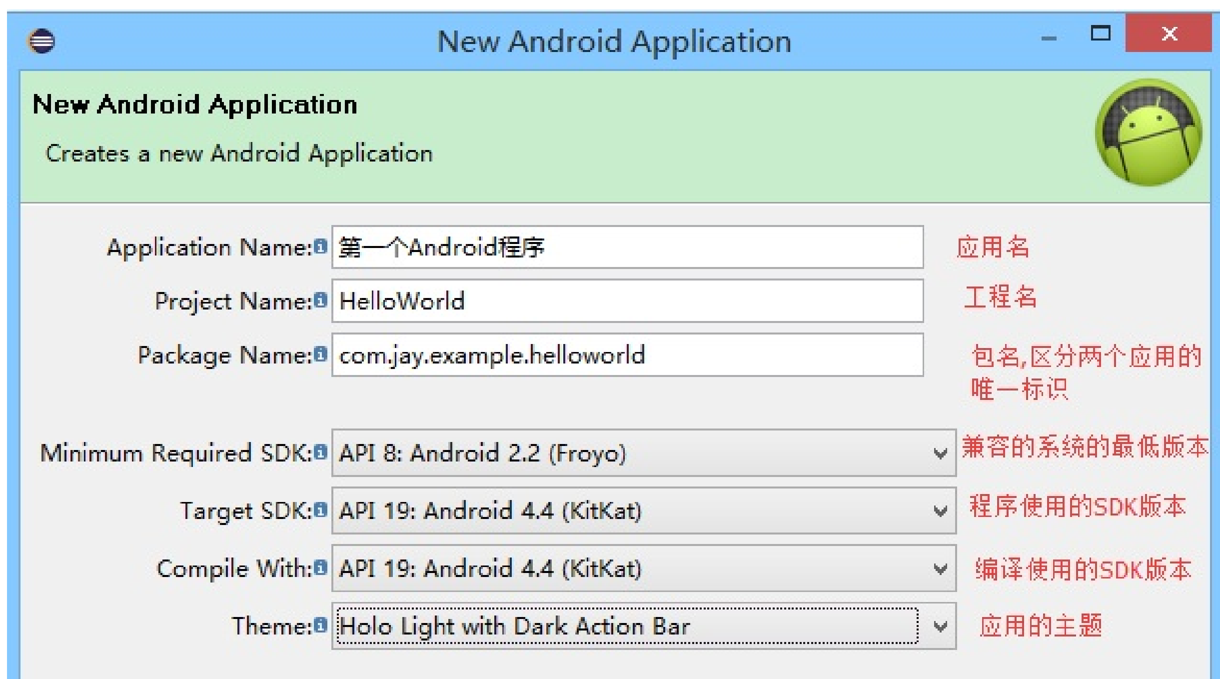
Eclipse可自行到Eclipse官网下载：Eclipse官方下载 而SDK和ADT可以到AndroidDevTools处下载：AndroidDevTools官网 这里给我们提供了很多Android开发相关的工具，而且不用翻墙，必备开发网站！务必Mark！ 不同版本配置间可能会有一些问题！笔者用的是旧版本的(很旧)，又需要的也可以下载：Eclipse：eclipse-jee-helios-win32.zip ADT：ADT-15.0.0.zip SDK：android-sdk-windows.rar 搭建流程： **1.**解压**Eclipse**：到解压的文件夹中找到eclipse.exe运行，运行后设置工程代码的存放位置(工作空间) **2.ADT**配置：依次点击菜单栏：**help -> Install new software -> Add -> Local... ->**选中下载加压后的**ADT**的文件夹 **->accept ->** 重启**Eclipse ->**看菜单栏是否出现**Android**小图标 如果出现表示安装完成 **PS**:期间可能出现一个waring,直接忽视~ **3.SDK**解压配置：依次点击菜单栏：**windows -> Rreferences -> Android ->**选中解压的**SDK**包 **-> OK ->**打开重新验证 **4.**创建**AVD(**安卓模拟器**)**：依次点击菜单栏：手机小图标 **-> New ->** 选定系统版本 **-> Skin**屏幕分辨率 **->**设置下**SD**卡大小 **->** 完成 **-> start**即可！ **PS:**第一次启动AVD的话可能很慢，需要等等~

4.第一个程序HelloWorld工程创建与运行

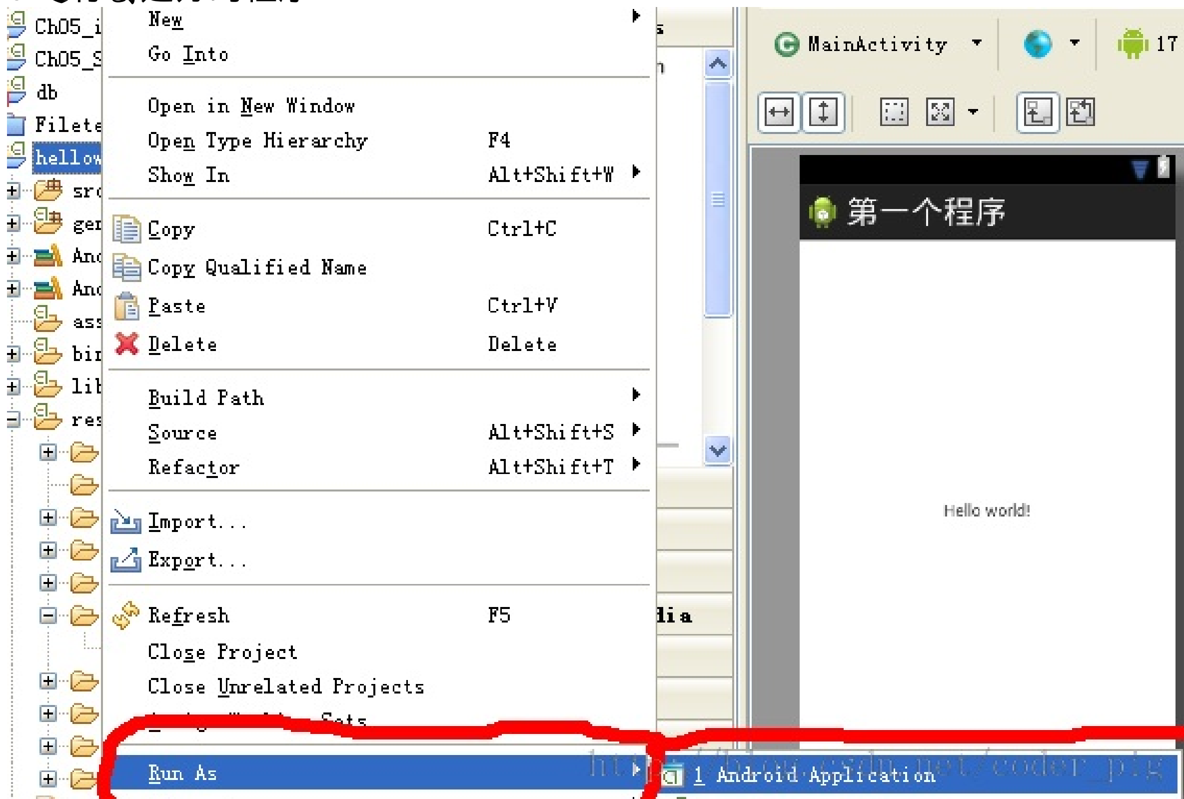
1.New -> Android App Project 如果找不到,可以去Other -> android找到，也是一样的：



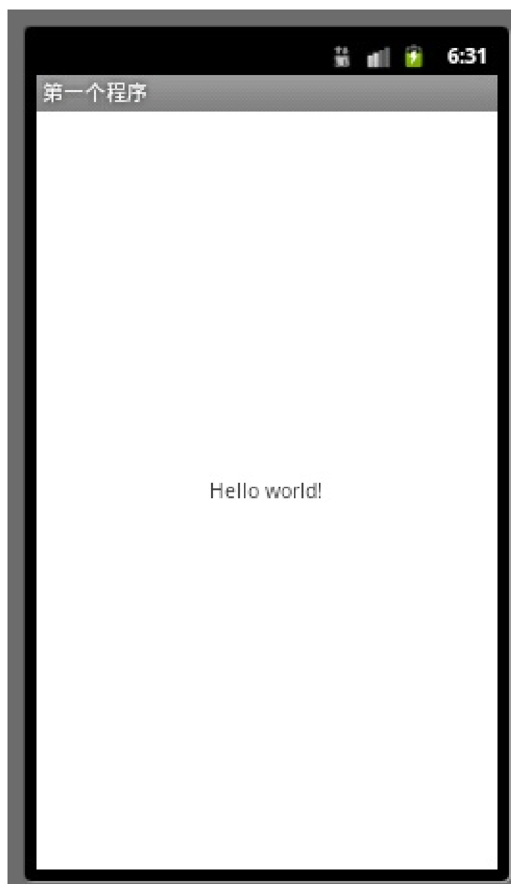
2.然后依次输入工程信息：



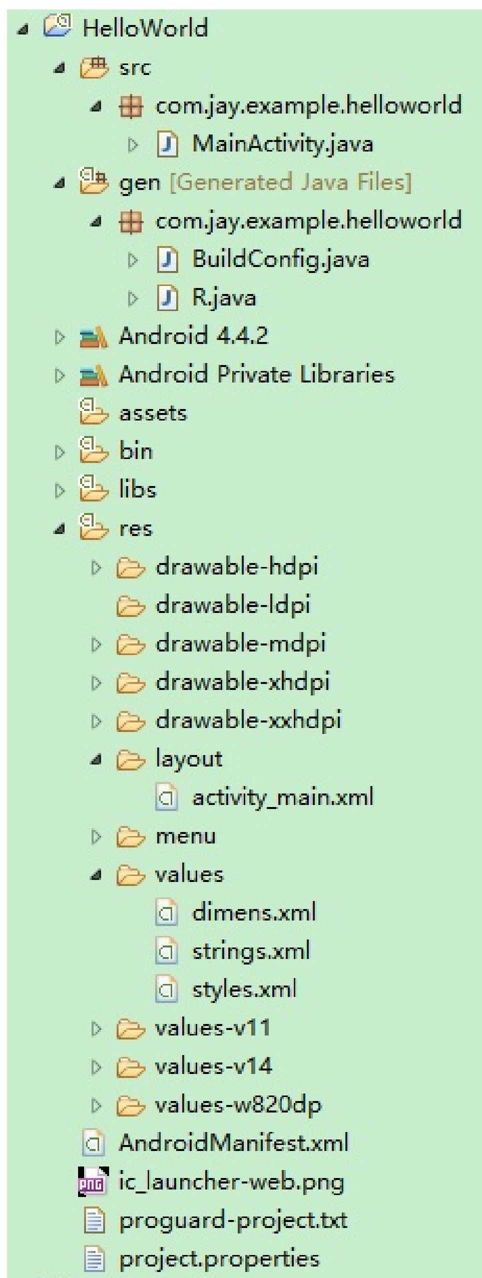
3.运行创建好的程序



4.从

模拟器上看到运行效果：

5.项目的目录结构分析

先来看下我们的工程目录图：

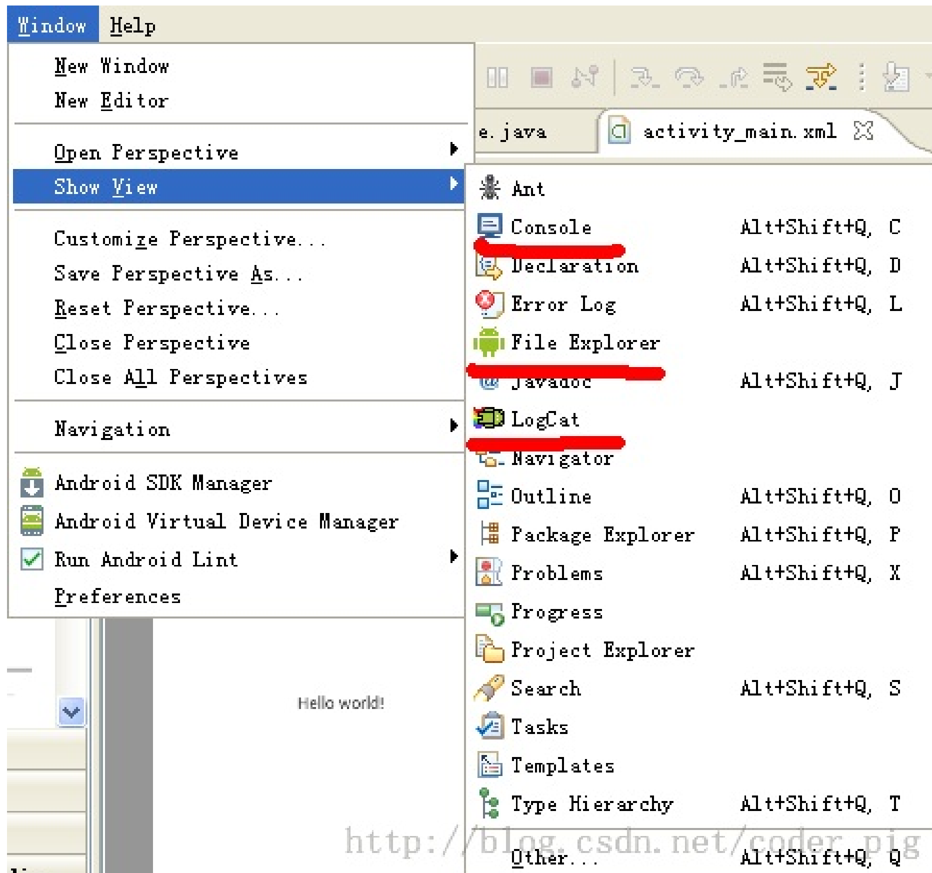
接下来我们需要知道的部分：

- **src**目录：包含App所需的全部程序代码文件，我们大多数时候都是在这里编写我们的Java代码的
- **gen**目录：只关注R.java文件，它是由ADT自动产生的，里面定义了一个R类，可以看作一个id(资源编号)的字典，包含了用户界面，图形，字符串等资源的id，而我们平时使用资源也是通过R文件来调用的，同时编译器也会看这个资源列表，没有用到的资源不会被编译进去，可以为App节省空间
- **assets**目录：存放资源，而且不会再R.java文件下生成资源id，需要使用AssetsManager类进行访问
- **libs**目录：存放一些jar包，比如v4,v7的兼容包，又或者是第三方的一些包
- **res**资源目录：存放资源的，drawable：存放图片资源；layout：存放界面的布局文件，都是XML文件； values：包含使用XML格式的参数的描述文件，如string.xml字符串，color.xml颜色，style.xml风格样式等
- **AndroidManifest.xml**配置文件：系统的控制文件，用于告诉Android系统App所包含的一些基本信息，比如组件，资源，以及需要的权限，以及兼容的最低

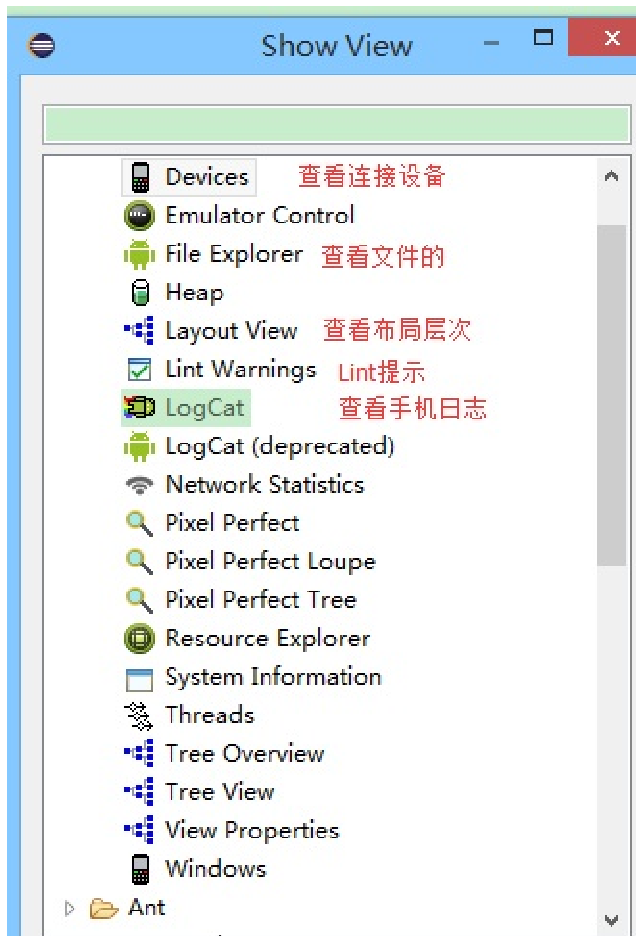
版本的SDK等

6.几个常用的视图

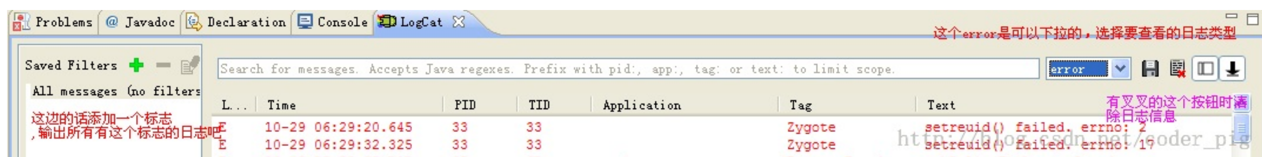
点击菜单栏上的:Windows -> show view打开对应的视图即可：



点击other,下述是Android中一些常用的视图:



其实主要的还是Logcat的使用，因为和Java不同，我们的App运行在虚拟机上，而我们的控制台却并不会显示相关信息，只有安装状态而已，所以我们会在Logcat上查看程序运行的日志信息：



7.本节小结

本节我们学习了使用Eclipse搭建我们的Android开发环境，懒人版和动手搭建版，两者都可以；接着我们又介绍了下Eclipse开发Android项目的目录结构的一些信息，在下一节中我们将深入HelloWorld工程，了解其中的代码！

1.2.2 使用Android Studio开发Android APP

写在前面

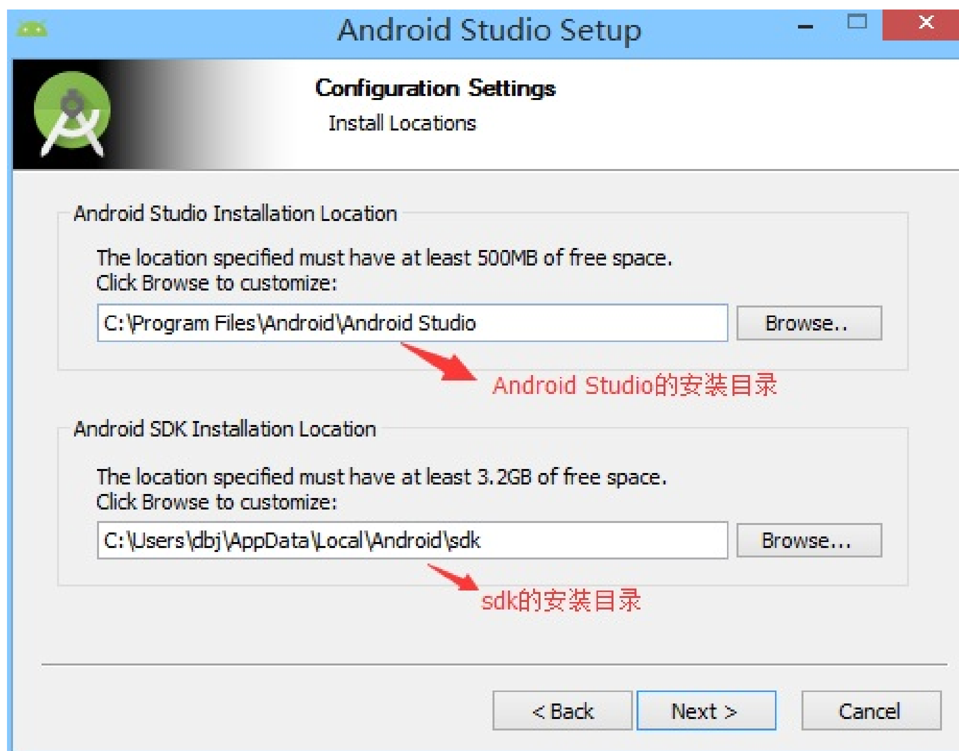
> 本节将介绍如何使用Android Studio开发Android APP，和前面Eclipse + ADT + SDK搭建Android开发环境一样，本节也只是介绍一些基本东西，深入的，比如快捷键，小技巧等会再另一篇文章中详细地介绍！

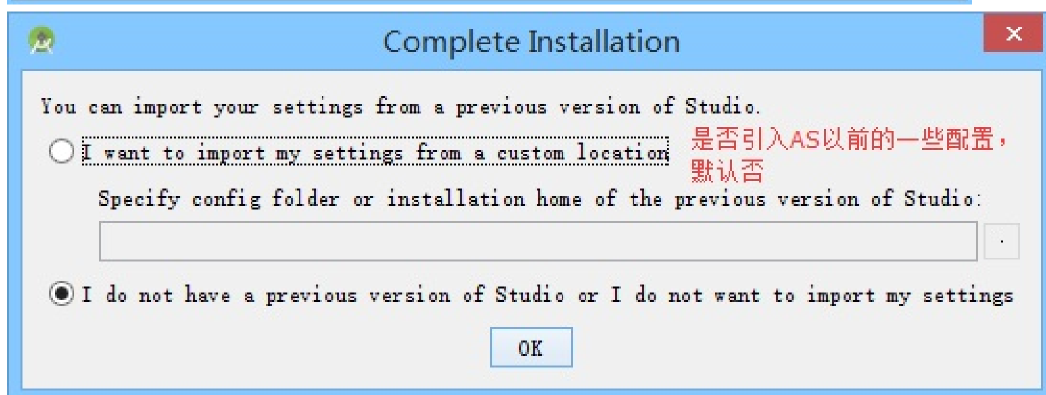
1.下载Android Studio

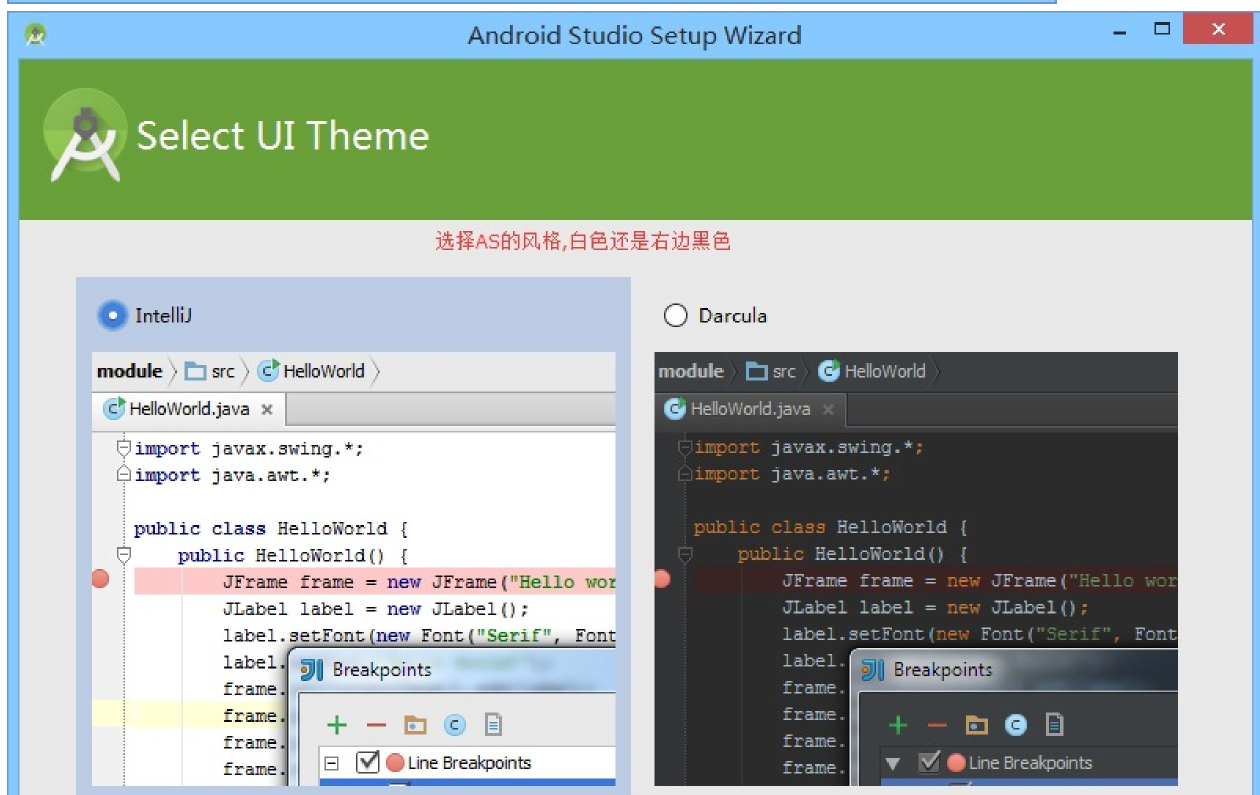
> 官网下载：Android Studio for Window ... 百度云下载：android-studio-bundle-141.1903250-windows.exe

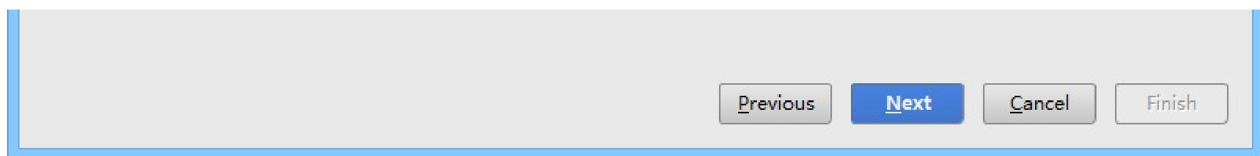
2.安装Android Studio

傻瓜式的下一步而已，只列出需要注意的页面：


Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
Android Studio的安装目录
Android SDK Installation Location
The location specified must have at least 3.2GB of free space.
Click Browse to customize:
C:\Users\dbj\AppData\Local\Android\sdk
Browse...
sdk的安装目录
< Back
Next >
Cancel


Complete Installation
You can import your settings from a previous version of Studio.
I want to import my settings from a custom location
是否引入AS以前的一些配置，默认否
Specify config folder or installation home of the previous version of Studio:
I do not have a previous version of Studio or I do not want to import my settings
OK

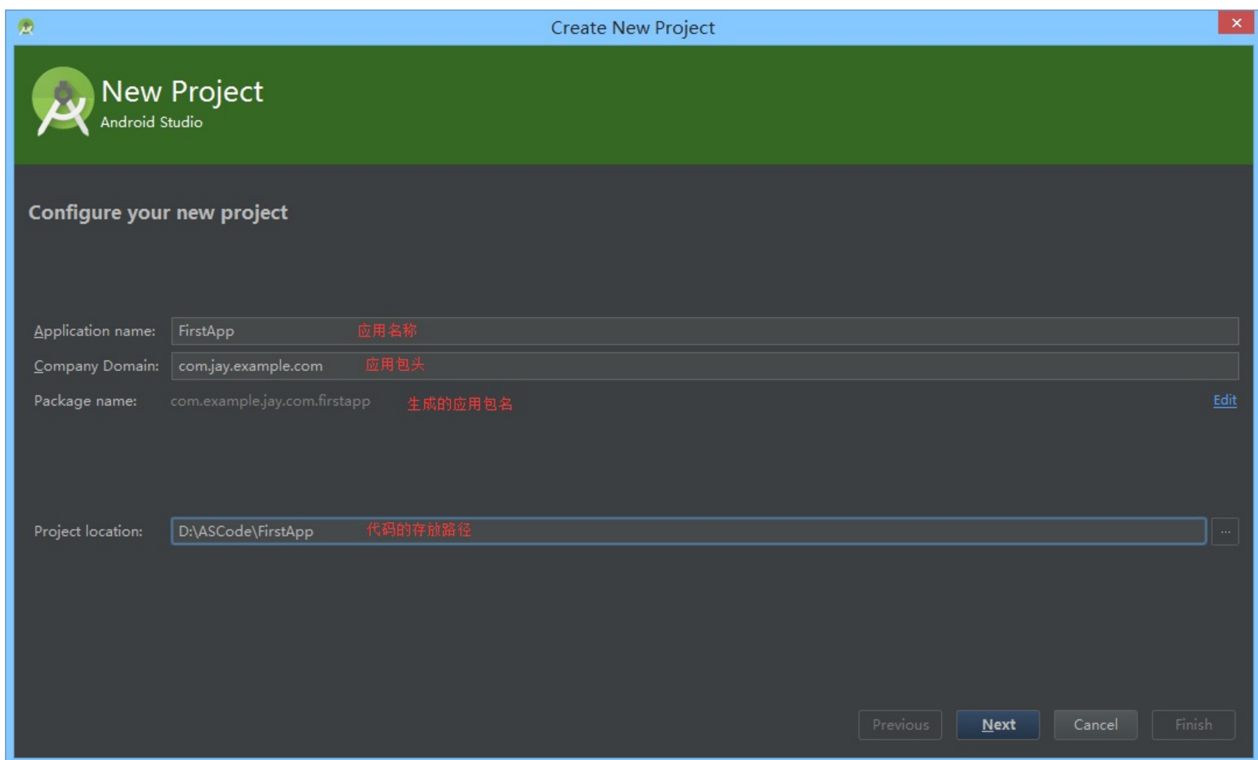

Android Studio Setup Wizard
Select UI Theme
选择AS的风格,白色还是右边黑色
IntelliJ
Darcula
module
src
HelloWorld
HelloWorld.java
Breakpoints
Line Breakpoints



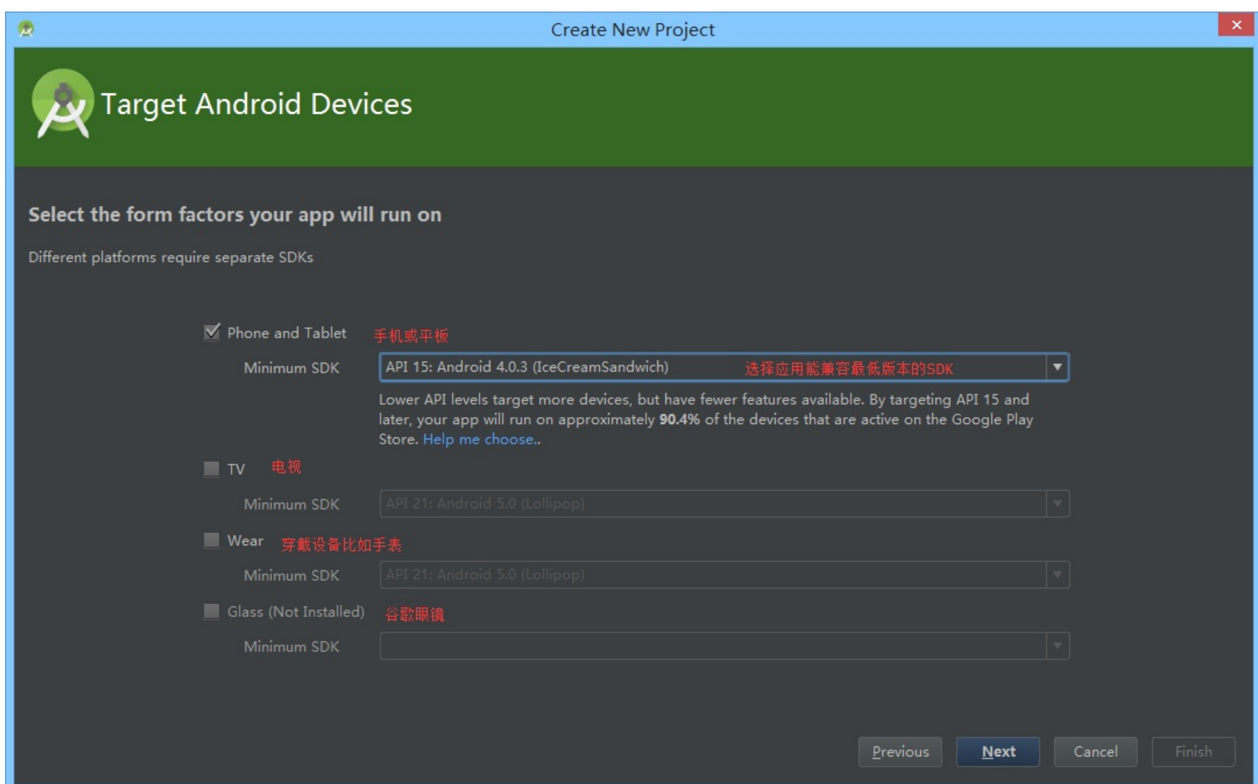
3.新建工程

安装完毕后，打开我们的Android Studio，第一次启动需要下载SDK等一些东西，时间比较长，笔者等了大概40分钟才下载完毕

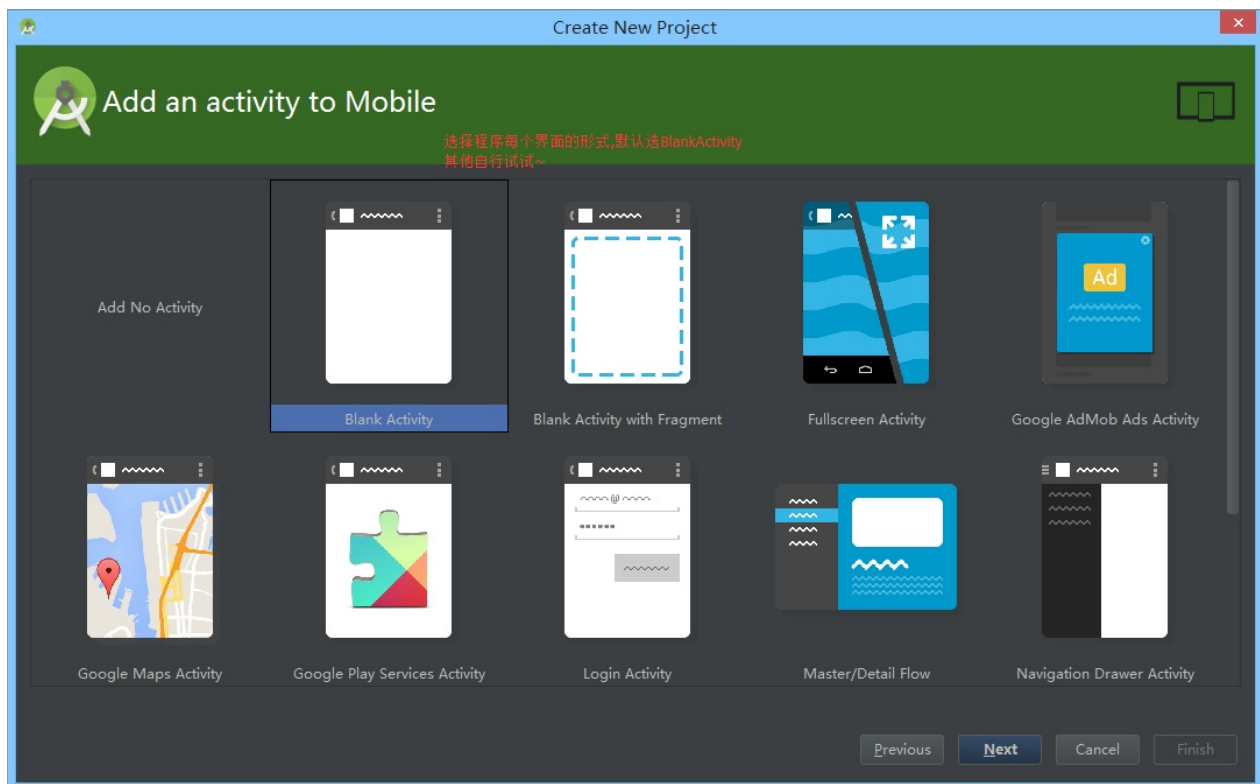
选择第一项,新建一个Android项目



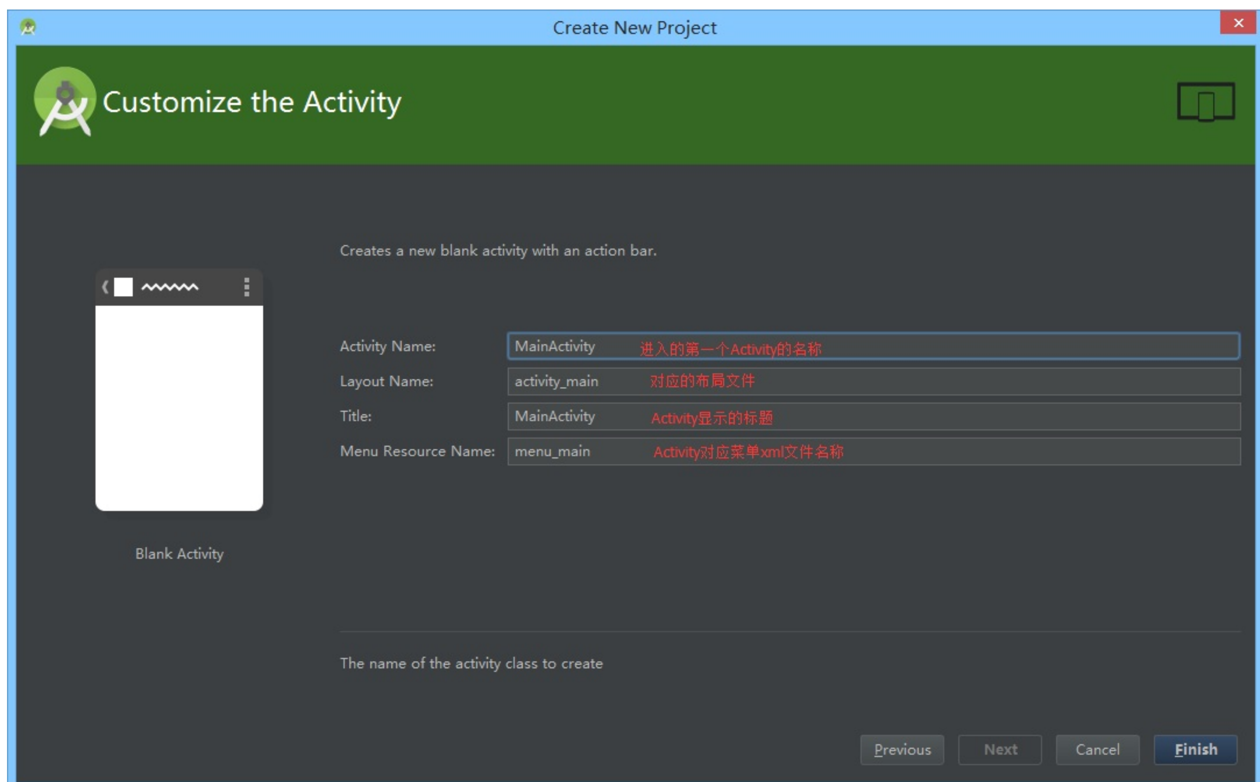
选择开发程序将运行在哪个平台上：



选择Activity的风格：



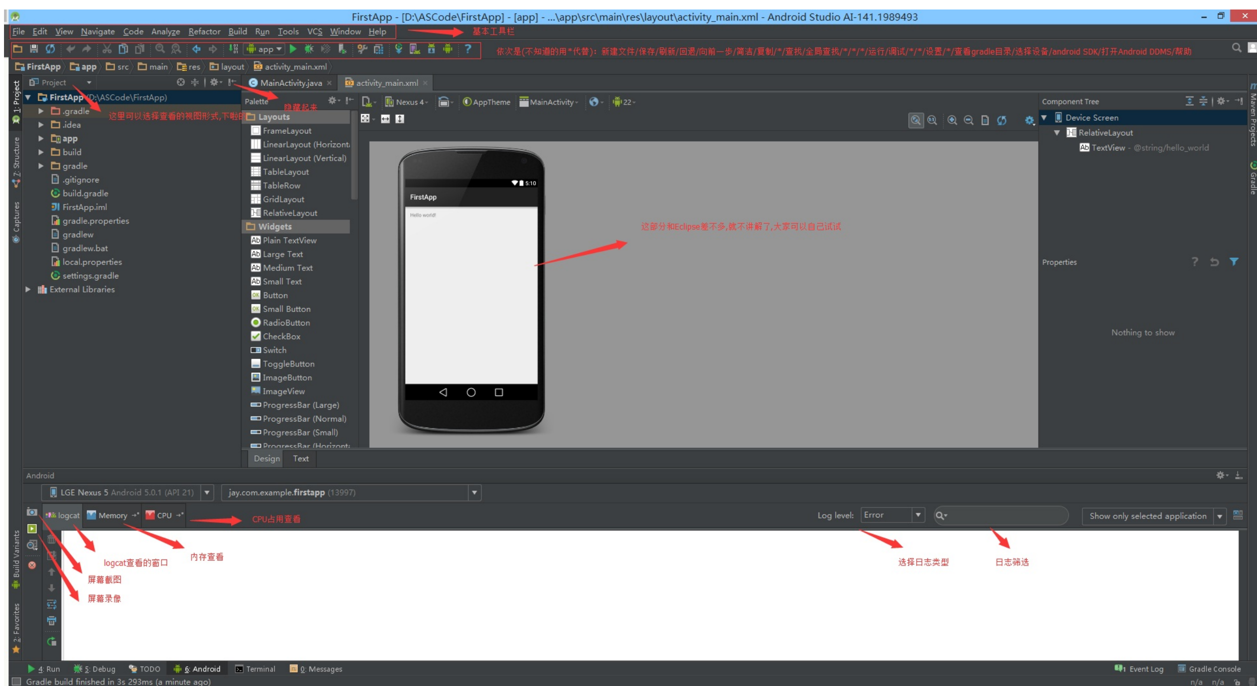
设置进入程序一个页面后的Activity的一些信息



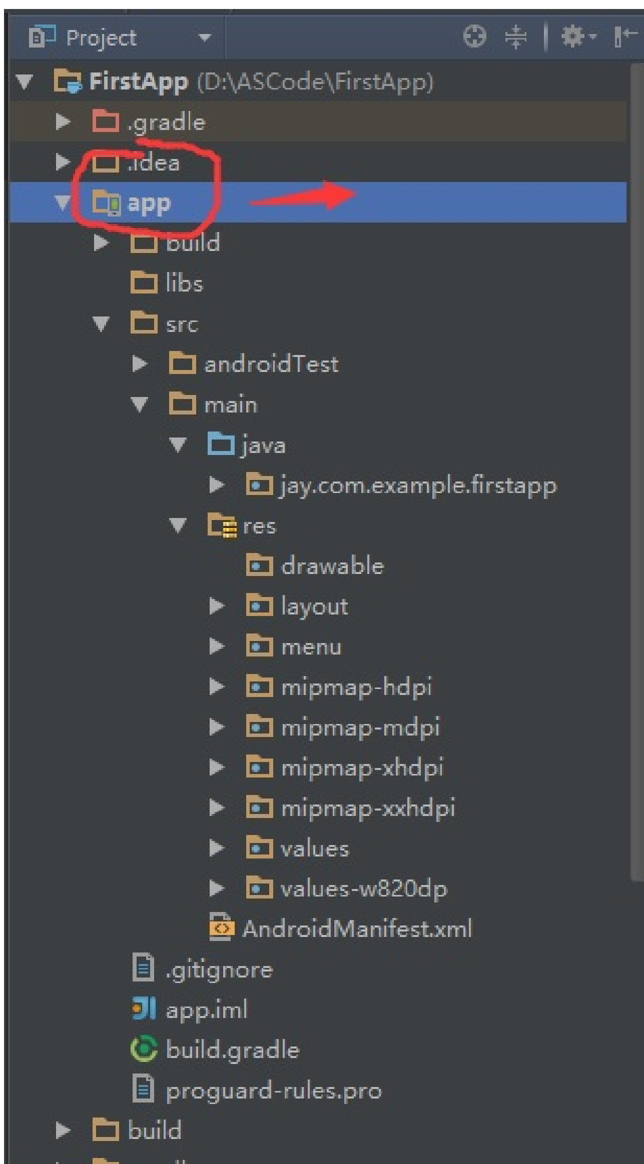
Finish然后是漫长的等待~

4.IDE的界面分析

先看下整个界面吧：



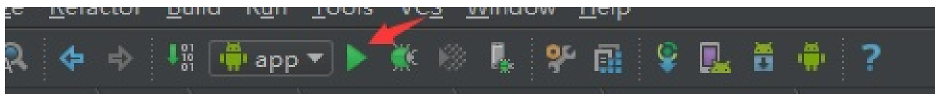
接着看下我们的项目结构，而我们一般关心的只是app这个目录：



build:构建目录,相当于Eclpise的bin目录
libs:依赖的包
src:
    androidTest:安卓单元测试的目录
    main
        Java: 写Java代码的地方
        res:资源文件
            drawable:图像资源
            layout:布局资源
            menu:菜单资源
            value:
                demens:css配置文件
                string:字符串资源
                styles:style资源
AndroidManifest.xml:配置文件
build.gradle:Gradle构建脚本

5.运行下程序试试

点击菜单栏的X，即可运行程序：



6.本节小结

Android Studio的用法还是比较简单的，当然笔者也是刚用Android Studio，后续还会写一篇更深入一点的文章，这里大家琢磨琢磨，暂时能跑起程序，知道在哪里写代码，怎么看Logcat就可以了！

1.3 SDK更新不了问题解决

问题阐述

> 相信大家在更新SDK的时候都会遇到更新不了的问题，而且打不开Google搜索，这是因为天朝墙了Google，所以要么只能通过VPN代理又或者改HOSTS才能访问，更新SDK！本节来介绍两种更新SDK的方法！

1.修改hosts文件

直接百度"Google hosts 2015"就有一堆了，而笔者常用的是：google hosts 2015,持续更新-360知识库 进去后复制分割线下所有的内容：

注4：google chrome 官方离线版本下载地址：传送门

------

google hosts 2015.06.11更新，本文只提供google相关服务的hosts，目前有效，失效后会及时更新

====更新分界线，复制下面内容到hosts文件即可=====

#google hosts 2015.06.11

64.233.162.81 google.com

64.233.162.81 www.google.com

64.233.162.81 m.google.com

64.233.162.81 scholar.google.com

64.233.162.81 translate.google.com

64.233.162.81 books.google.com.hk

从这里开始复制，知道底部

然后打开电脑上的这个路径：



用记事本打开，把复制的内容粘贴即可！

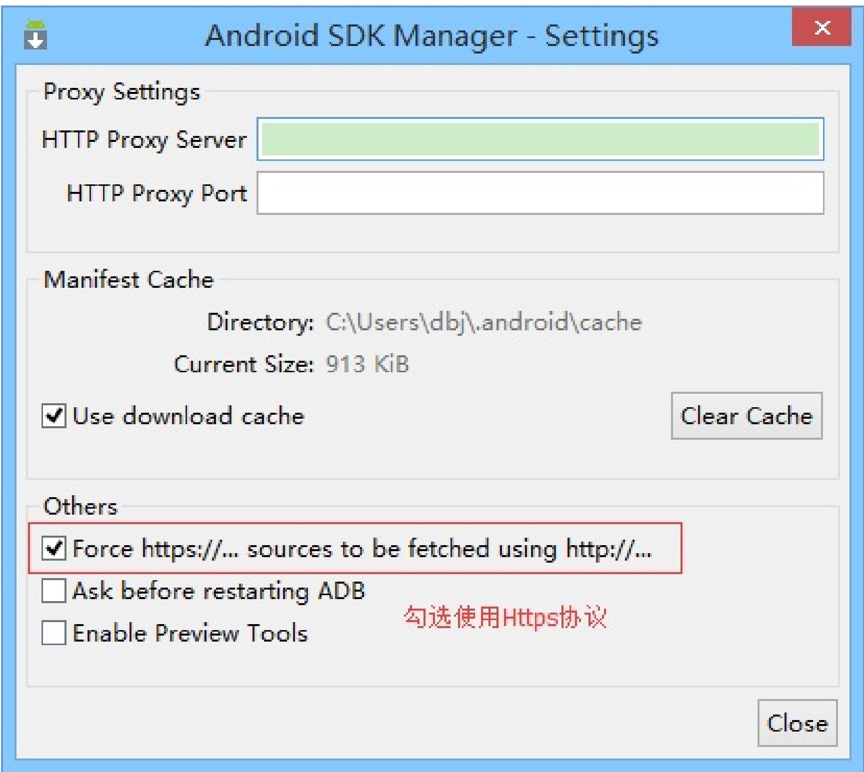
接着输入：https://www.google.com.hk/

出现：



表明修改Hosts成功

接着打开**sdk**设置下：



然后你会神奇的发现SDK可以更新了，而且你也能访问Google了，但是有点慢是吧，我们再介绍一个更新SDK的方法，速度快很多的！

2.使用国内镜像服务器更新

还记得前面给大家介绍一个Android开发必备的AndroidDevTools吗？打开这个网站，我们可以看到

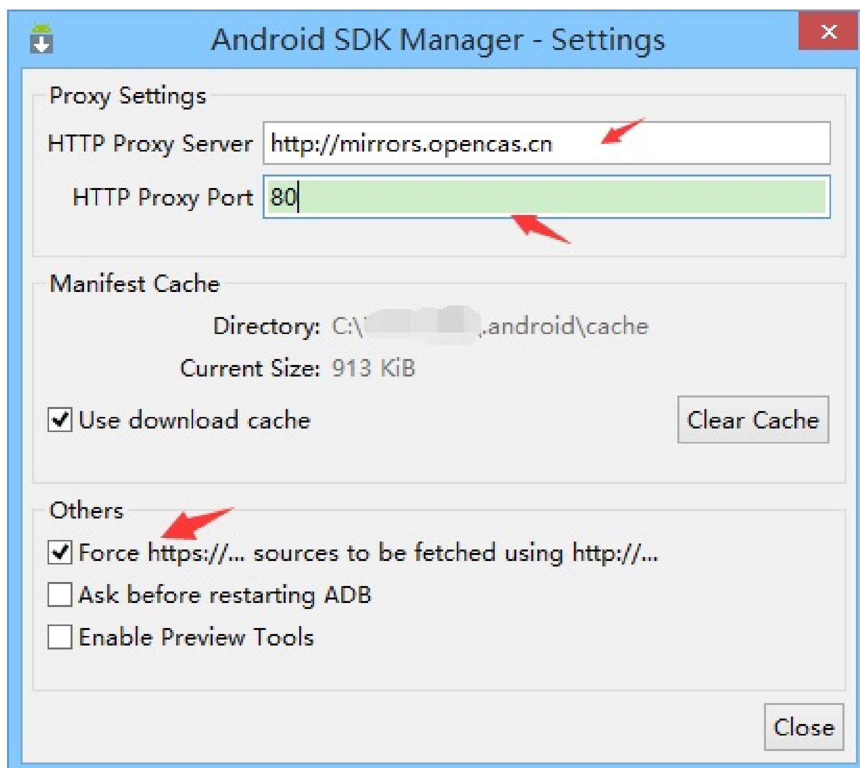
Android SDK在线更新镜像服务器

1. 中国科学院开源协会镜像站地址:
   - IPV4/IPV6: `http://mirrors.opencas.cn` 端口：80
   - IPV4/IPV6: `http://mirrors.opencas.org` 端口：80
   - IPV4/IPV6: `http://mirrors.opencas.ac.cn` 端口：80
2. 上海GDG镜像服务器地址:

   `http://sdk.gdgshanghai.com` 端口：8000
3. 北京化工大学镜像服务器地址:
   - IPv4: `http://ubuntu.buct.edu.cn/` 端口：80
   - IPv4: `http://ubuntu.buct.cn/` 端口：80
   - IPv6: `http://ubuntu.buct6.edu.cn/` 端口：80
4. 大连东软信息学院镜像服务器地址:

   `http://mirrors.neusoft.edu.cn` 端口：80

接下来我们只要选择上面随意一个，然后打开我们的Android SDK Manager，然后做如下设置： **Tools -> Option**，填入镜像源的地址和端号，勾选**Force https:**//使用Https协议



然后close，会到主界面，依次选择Packages -> Reload，就可以看到，刷刷刷进度条动了， 我们也可以选择对应版本的sdk进行下载了，而且速度还很快~

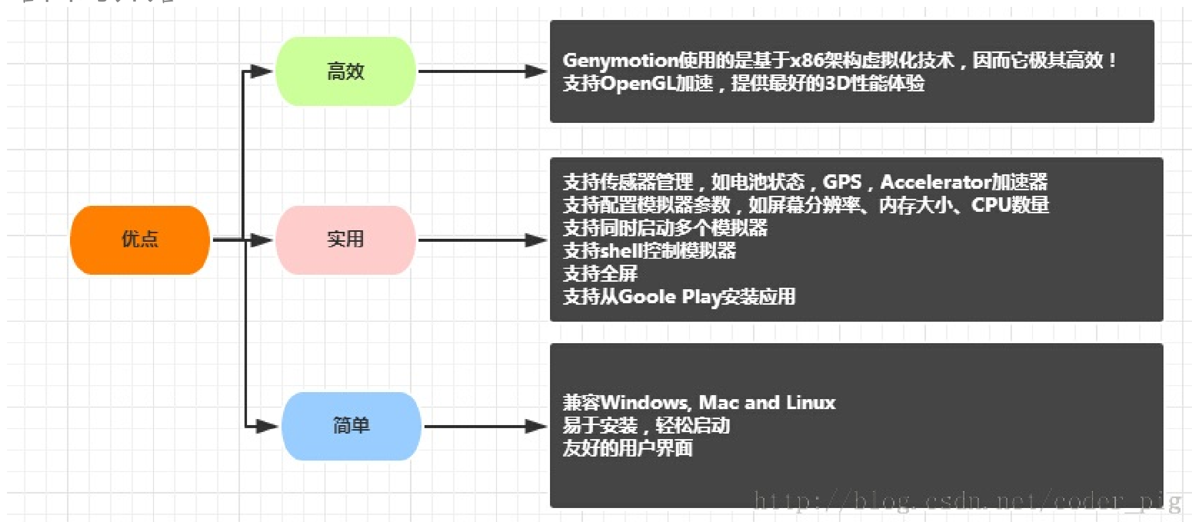
3.本节小结

本节给大家介绍了两种解决sdk更新不了的问题，改hosts的话是比较麻烦的，每隔几天就可能需要进行更新，如果仅仅是想更新sdk的话，建议使用国内镜像进行更新！

1.4 Genymotion模拟器安装

1.本节引言

> 如果你符合下述三种情况的话，你可以考虑安装一个Genymotion Android模拟器:
>
> 1. 没有真机调试，只能用模拟器
> 2. 嫌SDK内置的AVD启动速度，运行速度慢
> 3. 电脑配置还可以，最好4G内存以上
>
> 如果你满足上述三种情况的话，那么装个比真机还快的Genymotion吧！ 官方给出的介绍：



2.去哪里下Genymotion

> 百度"Genymotion"第一个就是了：Genymotion中文官网

3.下载Genymotion

> 点开上述链接后：点击注册



来到下述界面，如果已有账号，直接输入后**sign in** 如果没有的话，

Sign in

To use Genymotion, you must have a Genymotion account.

Username or e-mail address

Password

Remember me

Please specify a username or an e-mail address.

Sign in　Or　Create account

Forgot your password?

点击**Create account**创建一个新的账户

Account creation

2@qq.com

依次为账号,邮箱,密码

Your profile (optional)

使用人数:个人用免费,当然有些功能限制了

Company size Personal use

Usage type Development

使用Genymotion的目的,~开发

Allow Genymotion to send me e-mails about new releases 是否接受gm的邮件推送

I accept terms of the privacy statement 同意协议~

Create account



然后我们会收到一个激活邮件：
打开邮箱，点击激活账号，然后过一会儿会收到另一封创建成功的邮件

Welcome to Genymotion - User account activation ☆
发件人：**Genymotion** <genymotion-activation@genymobile.com>
(由 bounces+854998-1b4c-2393654902=qq.com@email.cloud.genymotion.com 代发)
时　间：2015年6月17日(星期三) 上午10:16 (UTC+02:00 赫尔辛基、开罗、雅典、伊斯坦布尔时间)
收件人：新的生活,新的开始 @qq.com>

Hi coder-pig,

Your user account with the e-mail address qq.com has been created.

Please follow the link below to activate your account. The link will remain valid for 15 days.
Click here 点击

You will be able to change your settings (password, language, etc.) once your account is activated.

If you have not requested the creation of a Genymotion account or if you think this is an unauthorized

This e-mail has been generated automatically. Please, do not reply to this message.

Genymotion - Registration completed ☆
发件人：**Genymotion** <genymotion-activation@genymobile.com>
(由 bounces+854998-1b4c-2393654902=qq.com@email.cloud.genymotion.com 代发)
时　间：2015年6月17日(星期三) 上午10:19 (UTC+02:00 赫尔辛基、开罗、雅典、伊斯坦布尔时间)
收件人

Hi coder-pig,

Congratulations, **your Genymotion account has been created successfully** and we are pleased to
We recommend you keep this e-mail to store your credentials.

Your credentials:

- **Username**
- **E-mail address:** @qq.com

You can now sign in to Genymotion with **your username and password**.

Thank you for your trust in our solutions,
Genymotion Team

This e-mail has been generated automatically. Please, do not reply to this message.

登陆账号后，点击**Get Genymotion:**

A faster Android emulator

Genymotion is the next generation of the AndroVM open source project, alread
2,500,000 developers. It's even easier to use and offers lots more features.

Get Genymotion 点击

选择Free版的下载,然后选择带virtual box虚拟机的下载

For personal use only

Download

Download Genymotion

Using Genymotion without a license is for private, non-professional purposes only. For more information, read the Terms and Conditions of Use.

› Purchase a license › How to install my license

不带virtual box虚拟机的

Get Genymotion (117.47MB) 带virtual box虚拟机的 Get Genymotion (without VirtualBox) (25.39MB)

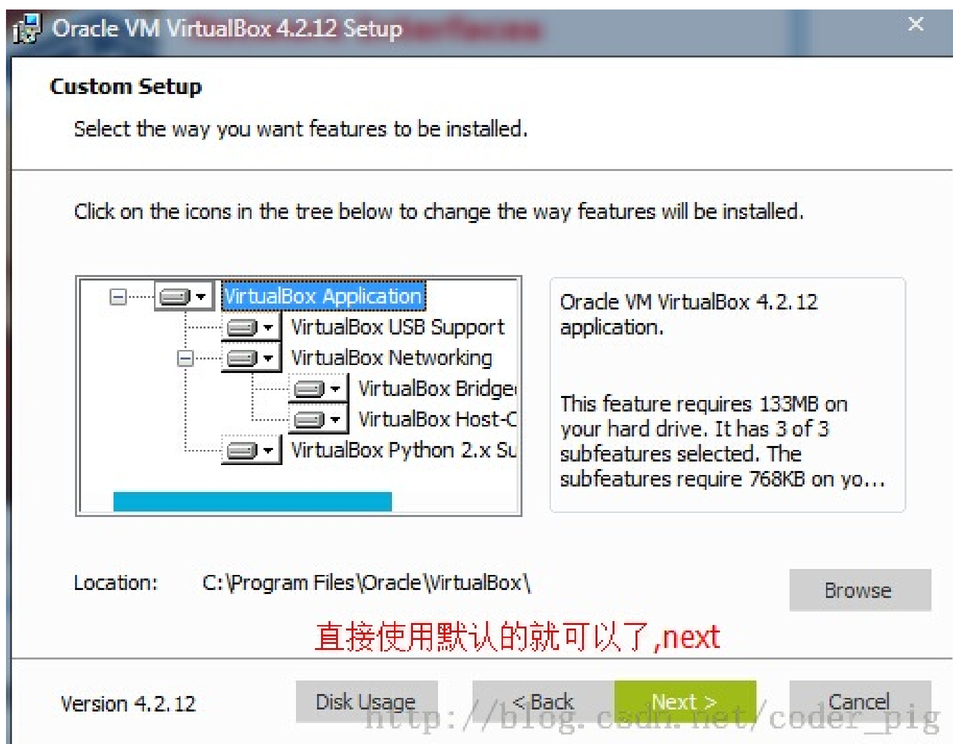
4.安装Genymotion

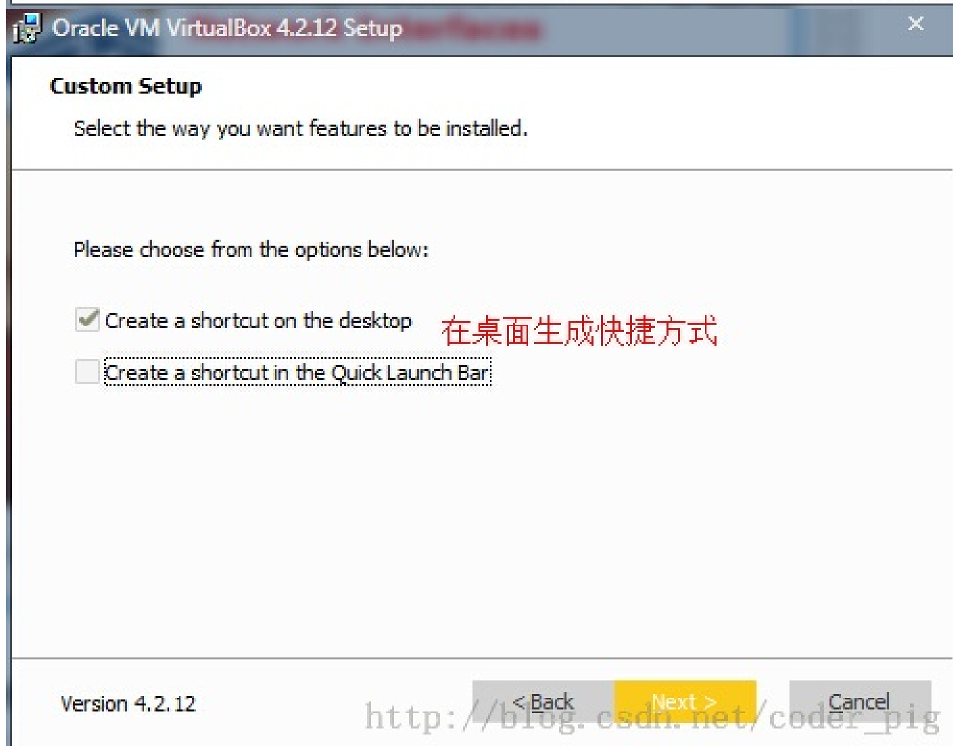
都是傻瓜式的下一步 选择安装目录：



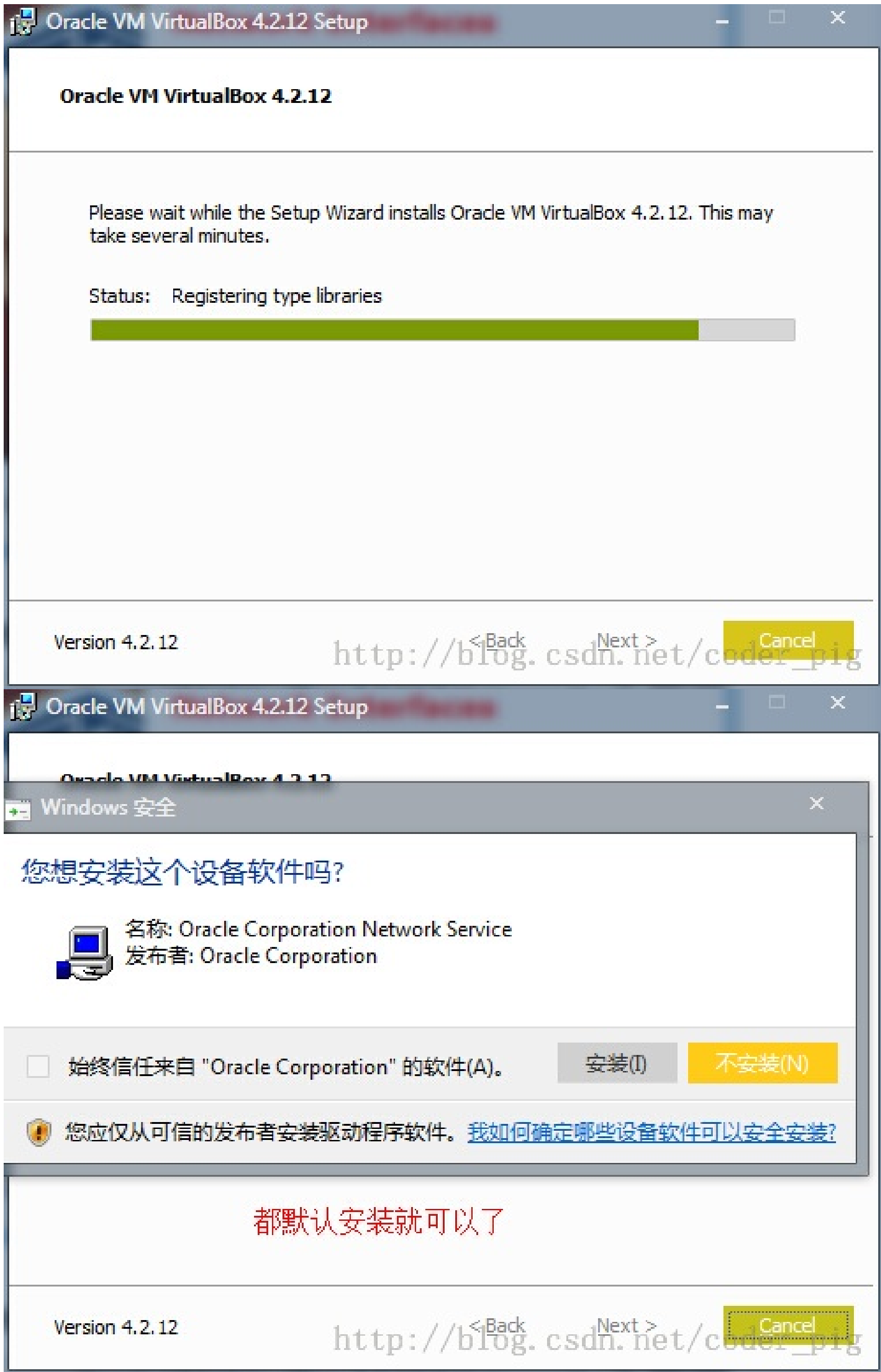
安装完后会弹出Oracle VM virtualBox的安装,这里可以选择路径,笔者直接默认安装了



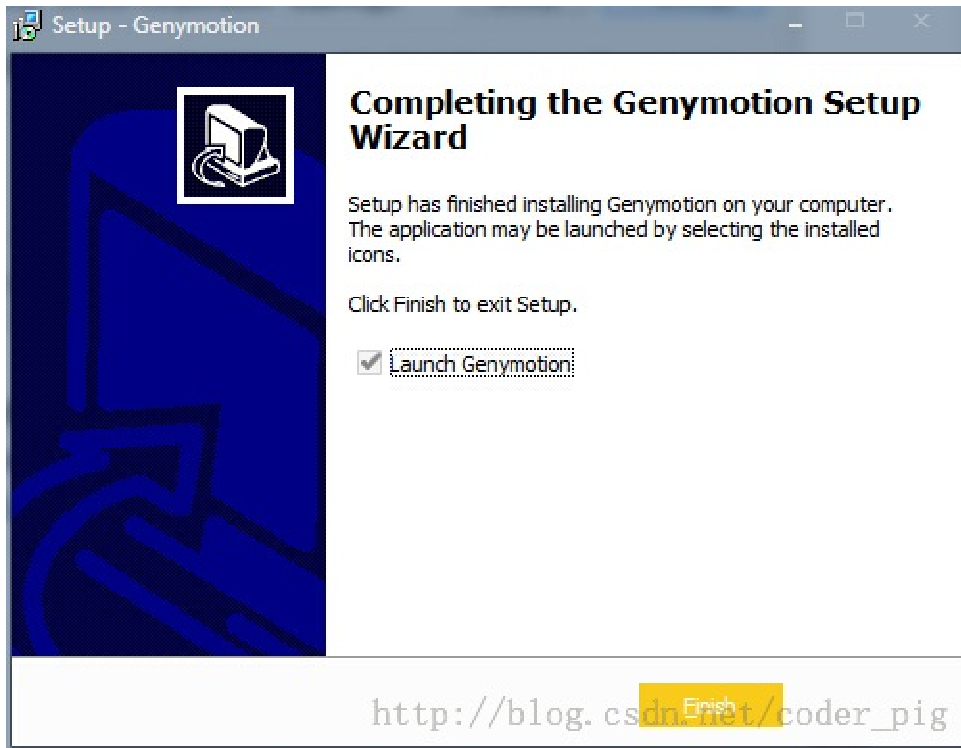




安装过程中会时不时弹出安装一些设备的窗口,这是虚拟机在安装东西,不用理,都按安装:

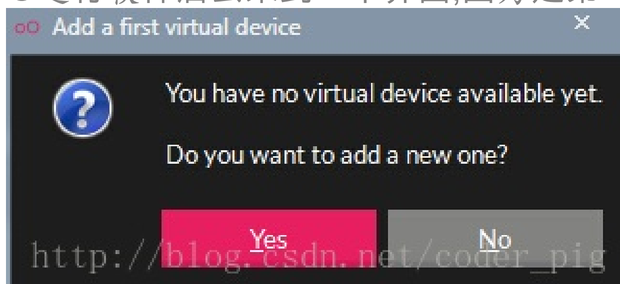


好了,安装完成后,确认,将我们的Genymotion运行起来

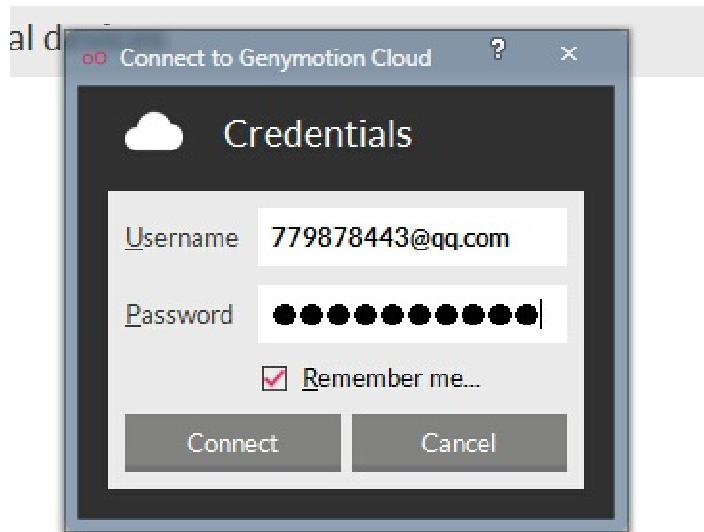


5.创建Android模拟器

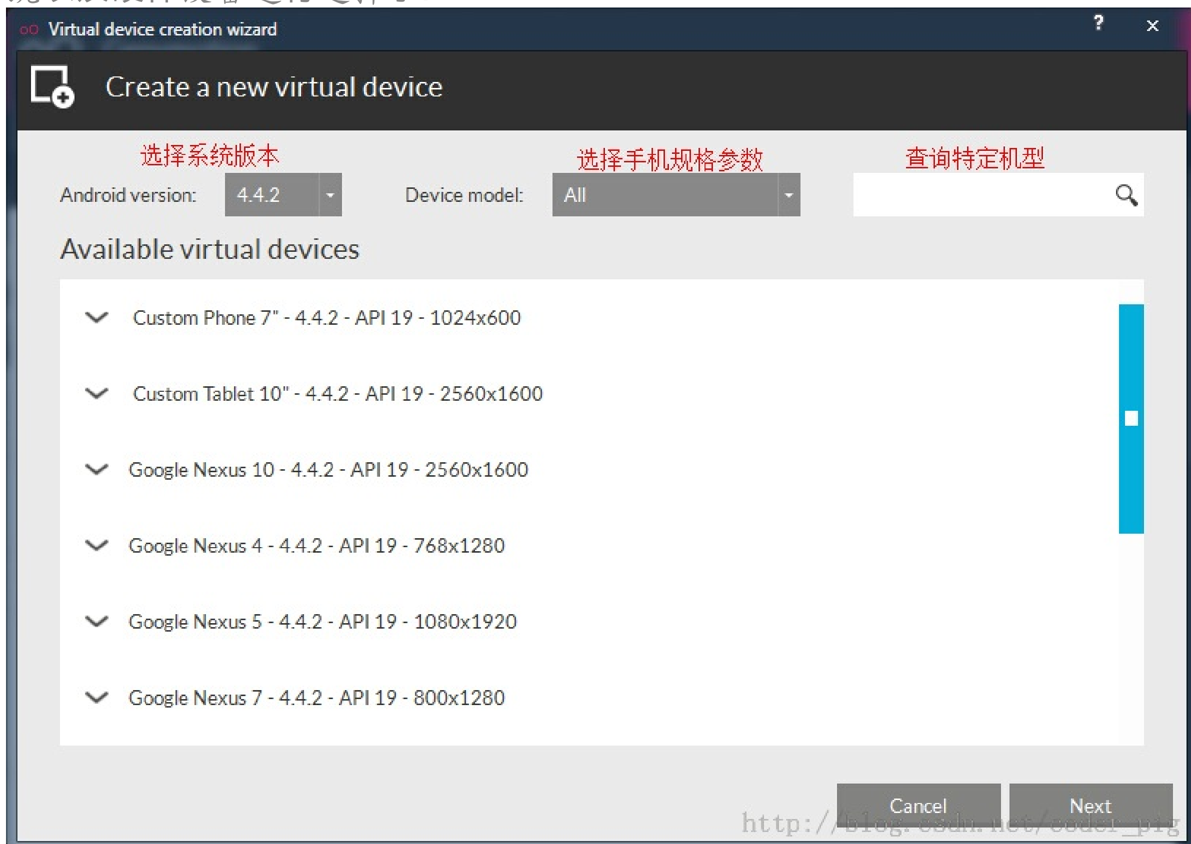
①运行软件后会来到一个界面,因为是第一次使用,所以需要我们新建一个avd:



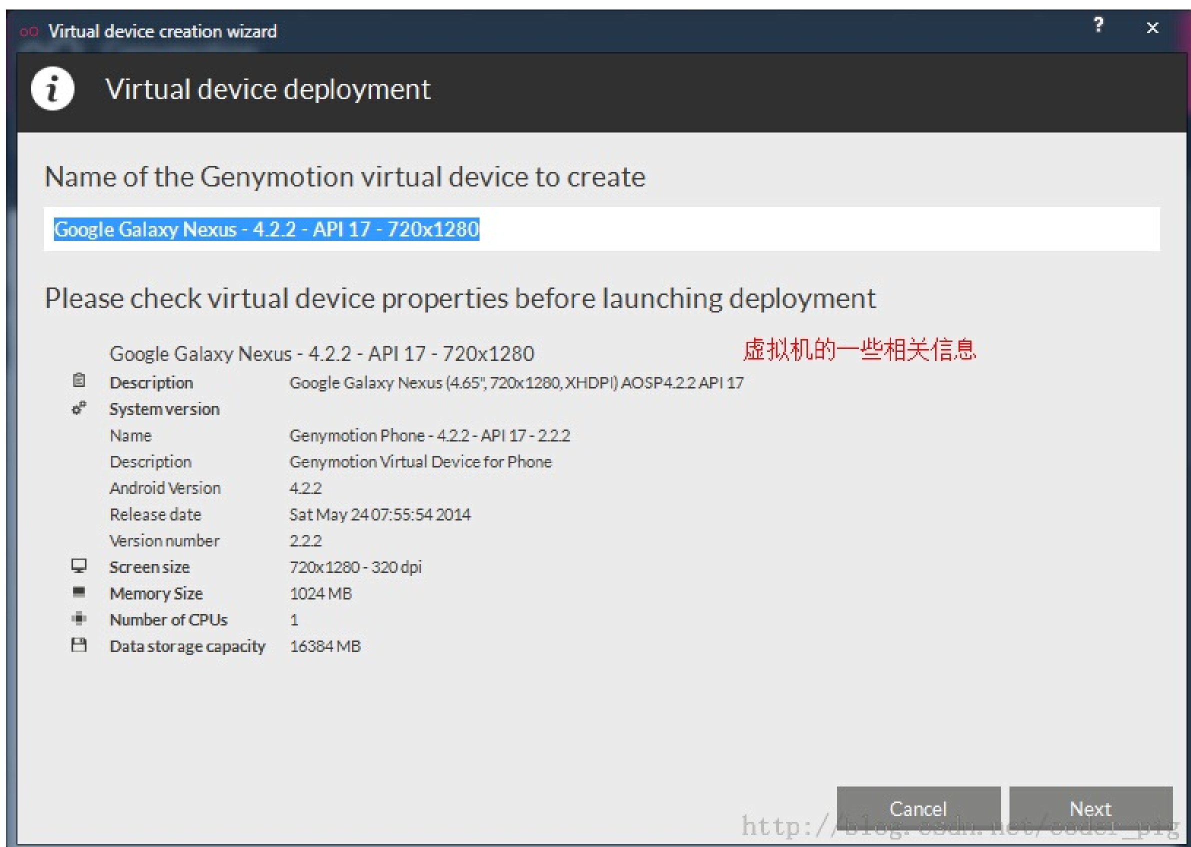
点击yes后会来到选择avd版本的界面,因为这上面什么设备都没有,我们需要连接到官网 获取各种版本的系统与硬件设备等,点击下方的connect,输入我们已经激活的账号密码: 邮箱与密码(笔者可能是网络问题,连续连了几次才成功的)



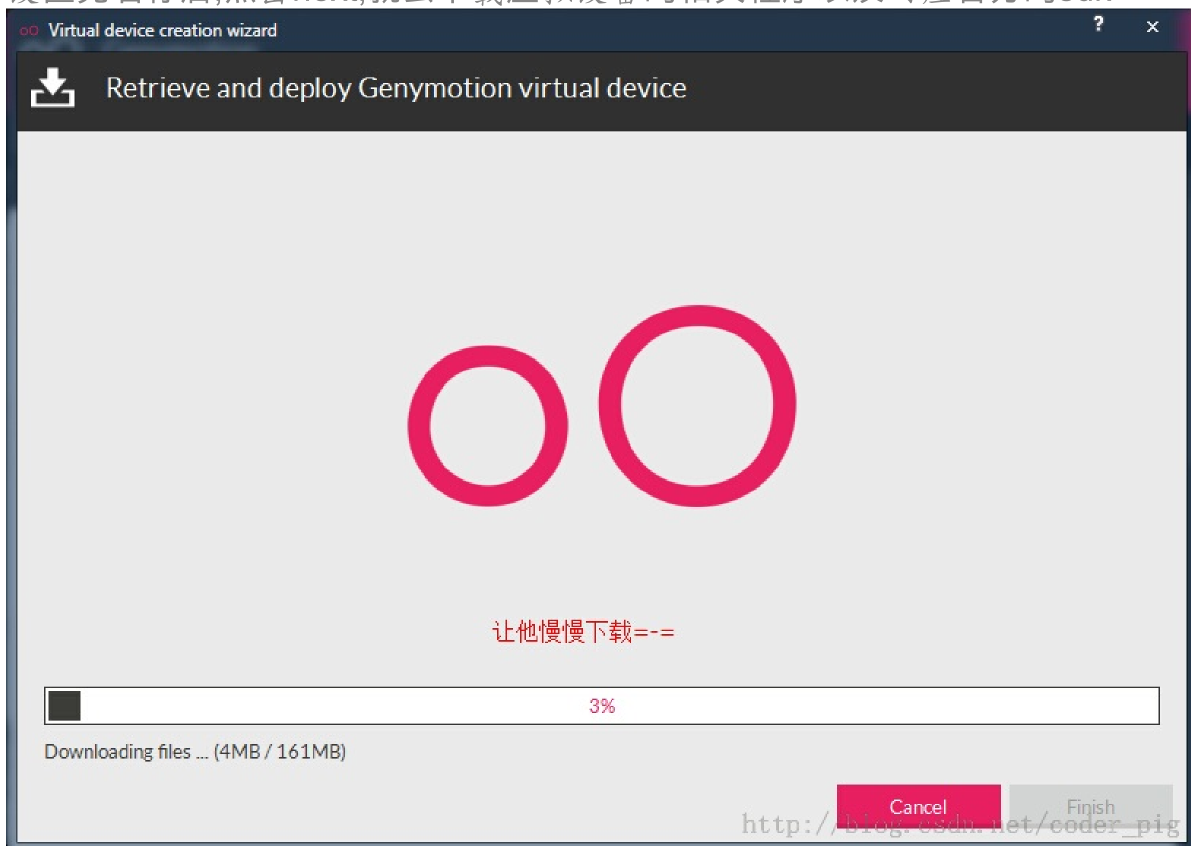
将注册时的账号密码填进去即可 登录成功后就可以对对应的系统以及硬件设备进行选择了:



这里显示设备的相关信息,可以自定义模拟器的名称



设置完名称后,点击next,就会下载虚拟设备的相关程序以及对应官方的sdk





好了,avd已经创建完毕,接着把他运行起来,开机也只是十几秒的事,很赞！操作起来超流畅啊,有木有？



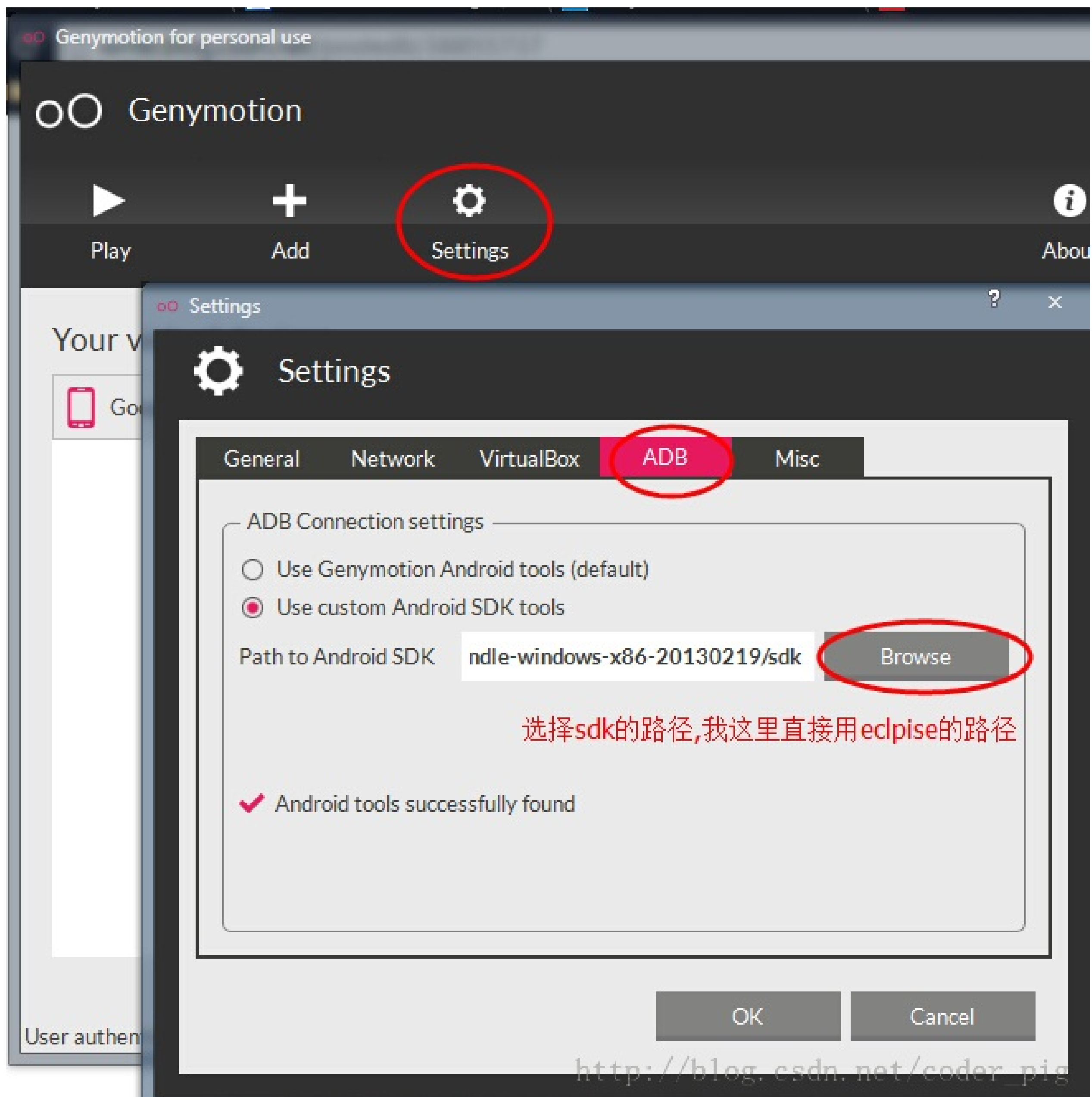
6.怎么在模拟器上运行程序

> 其实这个问题是白问的,我们只要在Eclpise上的Device就可以看到当前正在运行的模拟器; 感觉其实和真机是差不多的,我们只要运行程序时选择在哪个设备上运行即可！

7.可能遇到的问题

①不喜欢原生的系统,想用标准sdk中的系统版本

答:在下载设备驱动时其实已经下载好对应的sdk了,但是如果你不喜欢的话可以 打开setting--ADB-->选择Eclpise的sdk路径



②老登录不了?或者获取不了手机列表: 答:这个大部分的原因都是给墙了,所以只能用vpn了;不过貌似白天是可以的, 虽然有点慢,晚上的话笔者试了N次都是下载不了其他版本的设备!另外登录 不了要看下自己账号密码是否有错误哦！

③觉得模拟器占屏幕太大了 答:先把模拟器关了,点击扳手的图标



④运行程序，直接拖拉**APK**到模拟器上出现下述问题：

```
2014-09-06 11:33:20 - MyFinalFrame] Installation error: INSTALL_FAILED_CPU_ABI_INCOMPATIBLE
2014-09-06 11:33:20 - MyFinalFrame] Please check logcat output for more details.
2014-09-06 11:33:20 - MyFinalFrame] Launch canceled!
```

下载下面这个zip包,下载完毕后将它拖拽到模拟器窗口上,弹出对话框点击确定,接着重启下模拟器即可！ Genymotion-ARM-Translation.zip

⑤因程序需要，要查看模拟器的**sdk**目录： 之前的话我们在Eclipse上是通过mmt/sdcard找到sd卡目录的; 但是genymotion却不是在这个路径下:而是在下面这个路径下: /mnt/shell/emulated/0/ 可以根据后面的Info慢慢找出来

8.本节小结

本节介绍了比真机还快的安卓模拟器——Genymotion的安装与使用，相信各位读者会爱上这个模拟器的， 谢谢~

1.5.1 Git使用教程之本地仓库的基本操作

Git是什么？

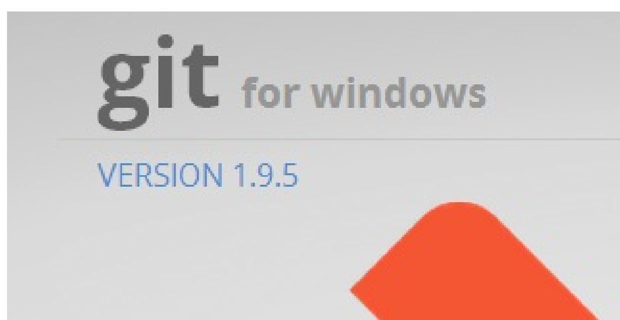
> 一个分布式版本控制系统，和SVN类似，但远比SVN强大的一个版本控制系统 ①Git可以方便的在本地进行版本管理，如同你本地有一个版本管理服务器一样 我们可以选择在合适的时间将本地版本推送到统一的版本管理服务器 ②Git每次会提取整个代码仓库的完整镜像，相当于对整个代码仓库都进行了一次备份，这样计时版本服务器除了问题，我们可以直接采用本地仓库恢复！结合本地版本管理功能，远程版本管理服务器出问题了，我们依然能继续写自己的代码，当他恢复的时候我们再提交我们的本地版本！ Git研发初期是为了更好的管理Linux内核，不过现在已经广泛应用于各种项目中！

安装Git

如果你的系统是Linux的话，直接打开shell输入:

```
sudo apt-get install git
```

当然，大部分的系统估计都是Windows，这就需要我们到网上下载一个Git For Window了，可到下述网站下载： http://msysgit.github.io/ 点击版本号，不是Download，不知道为什么打不开！



点击后进入页面，下载如下文件即可

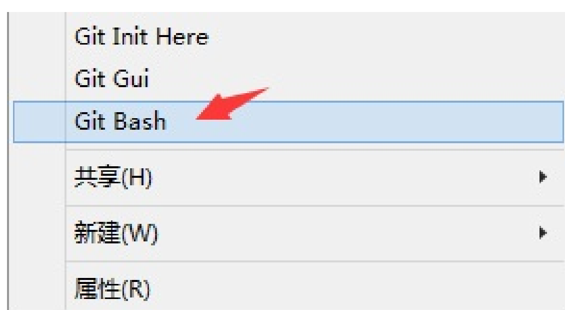
Downloads

Git-1.9.5-preview20150319.exe	17.1 MB
msysGit-netinstall-1.9.5-preview20150319.exe	2.92 MB
PortableGit-1.9.5-preview20150319.7z	22.5 MB
Source code (zip)	
Source code (tar.gz)	

或者到笔者的云盘直接下载也可以：Git-1.9.5-preview20150319.exe 接着傻瓜式的下一步就可以了~ 接下来你可以找到Git Gui然后开始玩Git，不过如果以后换到其他平台上，没有图形化界面你就寸步难行了！So，如果你有兴趣的话，我们来玩命令行，以后换了系统也能正常的玩Git！

玩转Git命令行

当然Git肯定是搭配着GitHub玩才够味的，不过先来学习一些本地的指令先把！当你安装完Git后我们可以在任意位置右键，点击Git bash打开我们的Git命令行！你可以可以点击Git Init Here直接在当前目录下创建一个代码仓库，又或者点击Git Gui打开Gui的图形操作页面！



1.创建代码仓库

Step 1：先配置下我们的身份吧，这样在提交代码的时候Git就可以知道是谁提交的，命令如下：

```
git config --global user.name "coder-pig" git congif --global user
```

配置完成后，我们可以再次输入，不包括名称，可以看到我们已经配置成功了

```
/D/新建文件夹
$ git config --global user.name "coder-pig"

/D/新建文件夹
$ git config --global user.name
coder-pig

/D/新建文件夹
$ git config --global user.email "779878443@qq.com"

/D/新建文件夹
$ git config --global user.email
779878443@qq.com
```

Step 2：找个地方创建我们的代码仓库，然后我创建了一个新的项目：TestForGit，来到工程的目录下，右键，打开我们的Git Bash，键入下述指令完成代码仓库的建立！另外这个代码仓库其实是用来保存版本管理所需的一些信息，我

们本地提交的代码都会提交到代码仓库中，于是乎我们可以选择还原到某个版本，当然，如果需要的话，我们还可以将保存在代码仓库中的代码推送那个到远程仓库中！比如GitHub!

```
git init
```

一个简单的代码，代码仓库就创建完毕了！继续输入：ls - al可以看到下目录下有个.git的文件夹就是他了！

```
/D/EACode/TestForGit
$ git init
Initialized empty Git repository in d:/EACode/TestForGit/.git/

/D/EACode/TestForGit (master)
$ ls -al
total 36
drwxr-xr-x    1 dbj      Administ     4096 Jun 19 15:31 .
drwxr-xr-x    1 dbj      Administ     4096 Jun 19 15:27 ..
-rw-r--r--    1 dbj      Administ      475 Jun 19 15:27 .classpath
drwxr-xr-x    1 dbj      Administ     4096 Jun 19 15:31 .git
-rw-r--r--    1 dbj      Administ      846 Jun 19 15:27 .project
drwxr-xr-x    3 dbj      Administ        0 Jun 19 15:27 .settings
-rw-r--r--    1 dbj      Administ      869 Jun 19 15:27 AndroidManifest.xml
drwxr-xr-x    1 dbj      Administ        0 Jun 19 15:27 assets
drwxr-xr-x    1 dbj      Administ        0 Jun 19 15:27 bin
drwxr-xr-x    1 dbj      Administ        0 Jun 19 15:27 gen
-rw-r--r--    1 dbj      Administ    51394 Jun 19 15:27 ic_launcher-web.png
drwxr-xr-x    1 dbj      Administ        0 Jun 19 15:27 libs
-rw-r--r--    1 dbj      Administ      781 Jun 19 15:27 proguard-project.txt
-rw-r--r--    1 dbj      Administ      563 Jun 19 15:27 project.properties
drwxr-xr-x    1 dbj      Administ     4096 Jun 19 15:27 res
```

也可以打开工程目录，同样看也看到.git文件夹；如果我们想删除代码仓库只需把这个文件夹删掉即可！

电脑 ▸ 文档 (D:) ▸ EACode ▸ TestForGit

名称	修改日期	类型
.git	2015/6/19 星期...	文件夹
.settings	2015/6/19 星期...	文件夹
assets	2015/6/19 星期...	文件夹
bin	2015/6/19 星期...	文件夹
gen	2015/6/19 星期...	文件夹
libs	2015/6/19 星期...	文件夹
res	2015/6/19 星期...	文件夹
src	2015/6/19 星期...	文件夹

2.提交本地代码

创建完代码仓库，接下来说下如何提交代码，我们是先用add命令把要提交的内容都加进来，然后commit才是真的去执行提交操作！命令例子如下，你可以一次次慢慢添加，当然也可以全部提交，直接git add .即可完成！ 我们现在工程目录下创建

一个readme.txt的文件试试，随便写点东西，然后依次输入下述指令：

```
git add readme.txt
git commit -m "Wrote a readme file"
```

输入命令试试：

```
/D/EACode/TestForGit (master)
$ git add readme.txt

/D/EACode/TestForGit (master)
$ git commit -m "Wrote a readme file"
[master (root-commit) 38f54a0] Wrote a readme file
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt

/D/EACode/TestForGit (master)
$
```

当然如果你可以add多个文件后再一次性commit，不过如果我们改动的文件很多的话，我们可以git add .一次添加全部，但有一些是几百年都不变一次的又或者自动生成的，比如lib，gen，bin文件夹等等，我们可以在代码仓库的根目录下创建一个名为.gitignore的文件，然后编辑里面的内容，把不需提交的文件忽略掉！

.git	2015/6/19 星期...	文件夹
.settings	2015/6/19 星期...	文件夹
assets	2015/6/19 星期...	文件夹
bin	2015/6/19 星期...	文件夹
gen	2015/6/19 星期...	文件夹
libs	2015/6/19 星期...	文件夹
res	2015/6/19 星期...	文件夹
src	2015/6/19 星期...	文件夹
.classpath	2015/6/19 星期...	CLASSPATH 文
.gitignore	2015/6/19 星期...	文本文档
.project	2015/6/19 星期...	PROJECT 文件

接着输入要提交时忽略的文件内容即可！

```
1  /gen/
2  /bin/
3  project.properties
```

那么我们git add .的时候，这里的文件就不会add，另外可能你会觉的commit后面写-m "xxx"很麻烦，想偷懒，但还是写上吧！输入的是本次提交的一些声明，比如自己修改了些什么！就好像写代码的时候，你偷懒不写注释，过几天你连自己写的什么鬼都不知道...

3.查看修改内容

好吧，前面我们用git add提交了整个项目到本地仓库，接下来我们改点东西，然后使用git status可以查看 修改的部分，比如，我们删掉MainActivity.java里的菜单的代码以及多余的菜单相关的包！

```
/D/EACode/TestForGit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/com/jay/example/testforgit/MainActivity.java

no changes added to commit (use "git add" and/or "git commit -a")
```

他就会提示我们哪些文件发生了改变，但是还没有提交，如果我们想看下具体更改了什么，我们可以用到git diff命令， 另外，按Q可以退回命令行输入！

```
/D/EACode/TestForGit (master)
$ git diff
diff --git a/src/com/jay/example/testforgit/MainActivity.java b/src/com/jay/exam
index 3181333..9946be2 100644
--- a/src/com/jay/example/testforgit/MainActivity.java
+++ b/src/com/jay/example/testforgit/MainActivity.java
@@ -2,8 +2,6 @@ package com.jay.example.testforgit;

 import android.app.Activity;
 import android.os.Bundle;
-import android.view.Menu;
-import android.view.MenuItem;

 public class MainActivity extends Activity {

@@ -12,23 +10,4 @@ public class MainActivity extends Activity {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
        }
-
-       @Override
-       public boolean onCreateOptionsMenu(Menu menu) {
-               // Inflate the menu; this adds items to the action bar if it is
-               getMenuInflater().inflate(R.menu.main, menu);
-               return true;
-       }
:
```

红色部分代表修改的部分

4.查看提交记录

当然随着我们项目的深入，Commit的次数也会越来越多，可能你早已忘记每次提交都修改了什么内容， 没事，Git帮你记着呢，使用git log即可查看历史提交信息！键入

```
git log
```

回车：

```
 /D/EACode/TestForGit (master)
$ git log
commit defd8af52be5183dfceb3e5cf23f78ea47d013b0
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 17:00:36 2015 +0800

    MainActivity Delete Menu

commit ad2080cf2ecfb8dd5bee3cbf0e10705f34bb64f8
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:43:01 2015 +0800

    All File Commit

commit 38f54a0205c2a439846a6a71ba9f0615c64c1de5
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:11:16 2015 +0800

    Wrote a readme file
```

我们取其中一小部分来分析：

```
commit defd8af52be5183dfceb3e5cf23f78ea47d013b0 Author: coder-pig <
```

依次是：

> - 此次提交对应的版本号
> - 提交人：姓名 邮箱
> - 提交的时间
> - 提交版本修改的内容：就是我们commit -m "xxx"里的xxx

5.撤销未提交的修改

比如我们刚提交了一个版本，然后又乱七八糟地写了一堆东西，突然发现不小心误删了一些东西，然后ctrl + s保存了，这个时候是不是欲哭无泪，不过有Git，只需一个checkout命令即可撤销更改，当然是你还没add的情况，比如我们在MainActivity里随便添加一条语句，然后ctrl + s保存代码！

```java
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        System.out.println("新写的东西");
    }
}
```

然后命令行键入：git diff：

```
/D/EACode/TestForGit (master)
$ git diff
diff --git a/src/com/jay/example/testforgit/MainActivity.java b/src/com/jay/exam
index 9946be2..cbbfe85 100644
--- a/src/com/jay/example/testforgit/MainActivity.java
+++ b/src/com/jay/example/testforgit/MainActivity.java
@@ -9,5 +9,6 @@ public class MainActivity extends Activity {
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
+               System.out.println("新写的东西");
 }
```

+号代表新增的内容，-好代表删除的内容

嗯，这里可以看到我们改的内容，我们可以回去把这句代码删掉，但是如果改的有上千行你怎么改，于是乎这个时候我们可以使用

```
git checkout src/com/jay/example/testforgit/MainActivity.java
```

然后会神奇的发现，我们新写的代码没了！duang一下就没了，不信你可以自己试试

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

当然，如果我们已经add了的话，那么checkout是没任何作用的，我们要先取消添加才可以撤回提交，使用下述指令：

```
git reset HEAD src/com/jay/example/testforgit/MainActivity.java
git checkout src/com/jay/example/testforgit/MainActivity.java
```

```
/D/EACode/TestForGit (master)
$ git reset HEAD src/com/jay/example/testforgit/MainActivity.java
Unstaged changes after reset:
M       src/com/jay/example/testforgit/MainActivity.java

/D/EACode/TestForGit (master)
$ git checkout src/com/jay/example/testforgit/MainActivity.java
```

6.版本回退

第五点我们教了大家撤销未提交的修改，但加入提交了，我们想回退到之前的某一个版本怎么办？第四点中我们可以通过git log查看我们的提交记录，我们需要从这里获取一个版本号，一般我们只需要前七位字符就够了；另外在Git中，用**HEAD**代表当前版本，上一个版本就是**HEAD^**，再上一个版本就是**HEAD^^**依次类推！我们先Git Log看下版本历史先！

```
/D/EACode/TestForGit (master)
$ git log
commit defd8af52be5183dfceb3e5cf23f78ea47d013b0
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 17:00:36 2015 +0800

    MainActivity Delete Menu

commit ad2080cf2ecfb8dd5bee3cbf0e10705f34bb64f8
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:43:01 2015 +0800

    All File Commit

commit 38f54a0205c2a439846a6a71ba9f0615c64c1de5
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:11:16 2015 +0800

    Wrote a readme file
```

我们回到前一个提交的版本吧，依次键入下述指令：

```
git reset --hard HEAD
git reset --hard HEAD^ git log
```

这时看下我们的控制台：

```
dbj@DBJTECH /D/EACode/TestForGit (master)
$ git reset --hard HEAD
HEAD is now at defd8af MainActivity Delete Menu

dbj@DBJTECH /D/EACode/TestForGit (master)
$ git reset --hard HEAD^
HEAD is now at ad2080c All File Commit

dbj@DBJTECH /D/EACode/TestForGit (master)
$ git log
commit ad2080cf2ecfb8dd5bee3cbf0e10705f34bb64f8
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:43:01 2015 +0800

    All File Commit

commit 38f54a0205c2a439846a6a71ba9f0615c64c1de5
Author: coder-pig <779878443@qq.com>
Date:   Fri Jun 19 16:11:16 2015 +0800

    Wrote a readme file
```

可以看到我们已经回退到了前一个版本了，当然你可以直接这样写：

```
git reset --hard ad2080c
```

就是这么简单！回退后，你突然后悔了，想回退回新的那个版本， 可是遗憾的是，你键入git log却发现没有了最新的那个版本号，这怎么办呢... 没事，Git中给你提供了这颗"后悔药"，Git记录着你输入的每一条指令呢！键入：

```
git reflog
```

你会发现，版本号就在这里：

```
 /D/EACode/TestForGit (master)
$ git reflog
ad2080c HEAD@{0}: reset: moving to HEAD^
defd8af HEAD@{1}: commit: MainActivity Delete Menu
ad2080c HEAD@{2}: commit: All File Commit
38f54a0 HEAD@{3}: commit (initial): Wrote a readme file
```

然后键入：

```
git reset --hard ad2080c
```

可以看到我们又回到了最新的那个版本了，就是这么溜！

```
 /D/EACode/TestForGit (master)
$ git reset --hard ad2080c
HEAD is now at ad2080c All File Commit
```

7.本节小节

> 本节给大家介绍了项目管理工具Git来管理我们的本地仓库，学习了一些基本的命令行操作，相信会给你的项目开发带来便利，当然本地远远是不够的，下一节我们将学习如何将我们的项目托管到GitHub上！敬请期待~

1.5.2 Git之使用GitHub搭建远程仓库

本节引言：

> 在上一节中，我们学习了如何使用Git，构建我们的本地仓库，轻松的实现了版本控制以及代码还原，修改日志查看等；读者肯定不满足与本地是吧，假如是多个人一起来开发一个程序呢？我们需要一个作为服务器的远程仓库！当然搭建一个服务器是需要成本的，为什么不把项目托管到Github上呢？作为开源代码库以及版本控制系统，Github拥有140多万开发者用户。随着越来越多的应用程序转移到了云上，Github已经成为了管理软件开发以及发现已有代码的首选方法，不需要任何成本，为何不使用呢？是吧！本节就来学习如何把我们的代码托管到Github上！

1.账号注册&仓库创建：

打开Github官网注册：Github官网，填写注册相关信息：用户昵称，邮箱，密码

ZPJay

qq.com

••••••••••

Use at least one lowercase letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.

注册完，跳转到如下页面，选择仓库购买方式(私有仓库,别人不可以访问，要权限)，一般我们自己玩选Free: PS:对了，这时候你邮箱可能收到一封验证邮件，点下完成验证。

Plan	Cost (view in CNY)	Private repositories	
Large	$50/month	50	Choose
Medium	$22/month	20	Choose
Small	$12/month	10	Choose
Micro	$7/month	5	Choose
Free	$0/month	0	Chosen

Your repositories 0 + New repository

You don't have any repositories yet!
Create your first repository or learn more about Git and GitHub.

接下来，创建一个我们的代码仓库：

Owner Repository name

ZPJay / Garbage

Great repository names are short and memorable. Need inspiration? How about potential-octo-archer.

Description (optional)

Test For Git

Public
Anyone can see this repository. You choose who can commit.

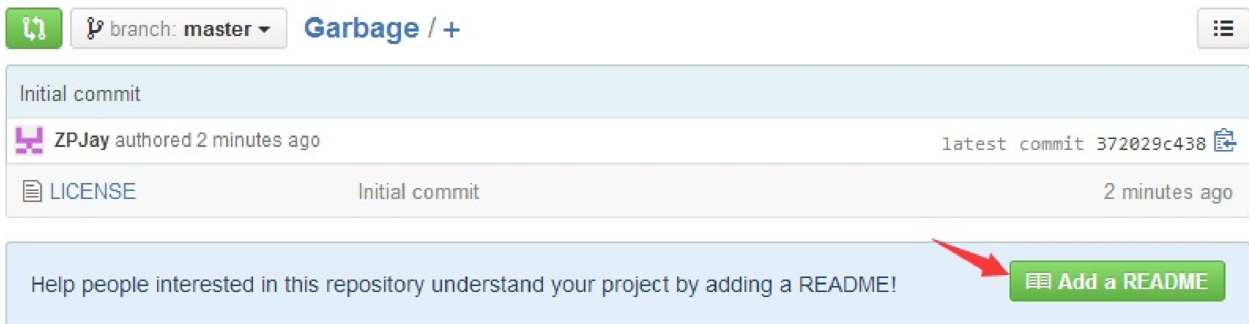
Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

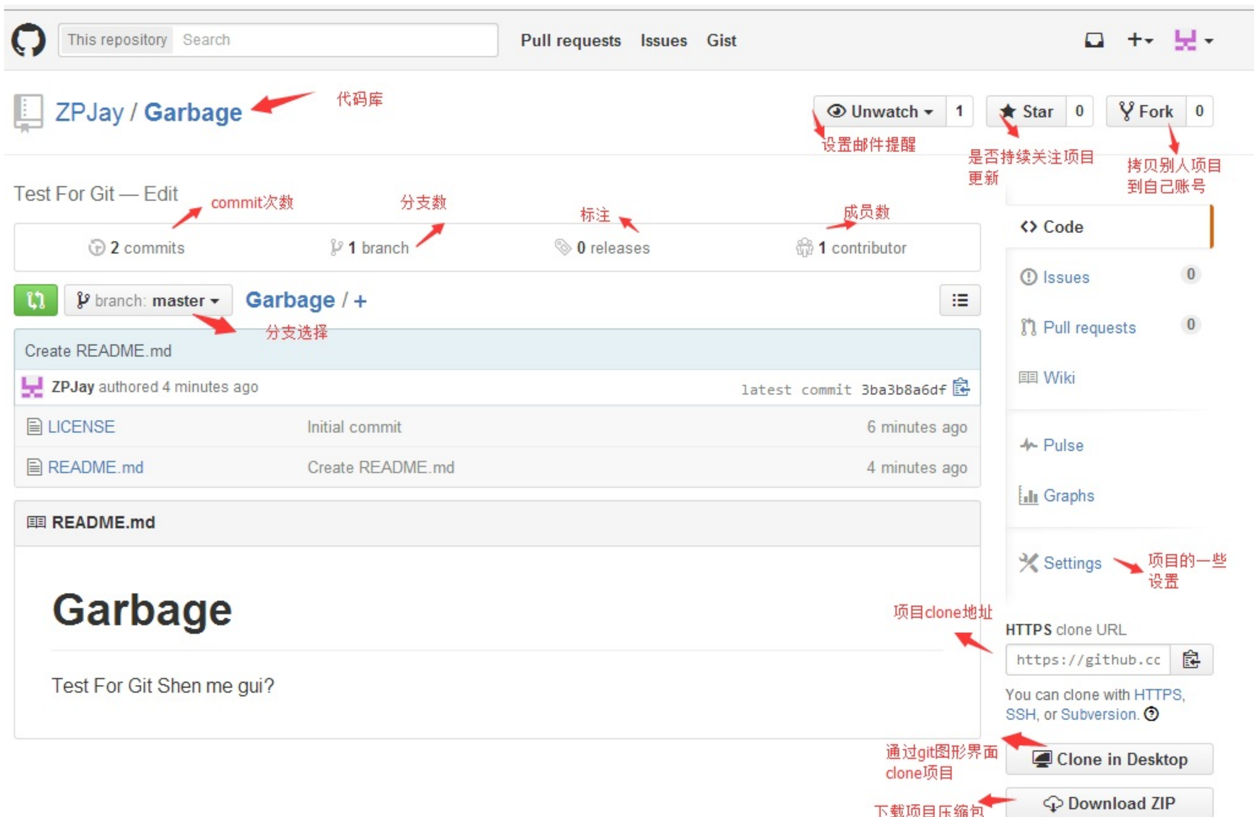
Add .gitignore: None | Add a license: Apache License 2.0

Create repository

为自己的仓库添加点内容提示，就是项目的一些概述(可写可不写)



简单介绍下主页的一些东西：

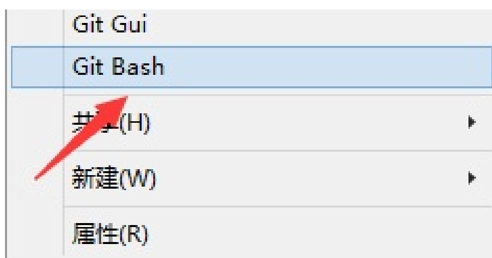


2.Clone代码库到本地

当然，你可以直接用图形化界面克隆，不过我还是喜欢通过命令行来Clone，先复制下Clone的地址



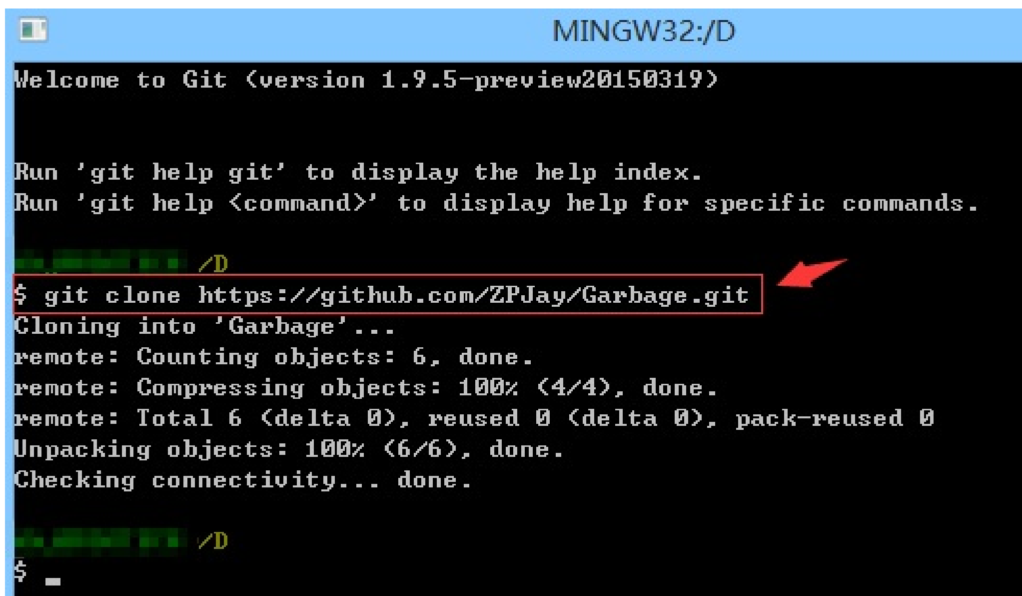
然后在某个地方，有键打开Git Bash：



键入：

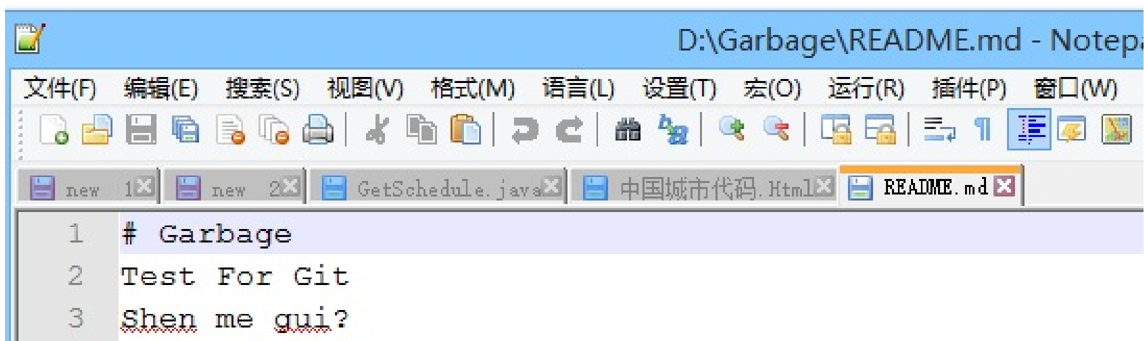
```
git clone https://github.com/ZPJay/Garbage.git
```

然后可以看到我们的代码库就下载完成了：



打开文件夹，可以看到下述内容：

.git	2015/6/24 星期...	文件夹	
LICENSE	2015/6/24 星期...	文件	12 KB
README.md	2015/6/24 星期...	MD 文件	1 KB



3.分支管理

> 对于刚接触版本控制工具的朋友来说，分支可能比较陌生，但是他会给我们带来很大的便利！限于篇幅， 笔者直接丢个链接，大家看看图就知道了：曹雪峰的官方网站：创建和合并分支！写得真心很赞~建议收藏！

了解概念后,我们来熟悉与分支相关的几个命令：

①创建分支(后者创建同时会切换分支):

```
git branch v1.0.3  或 git checkout -b v1.0.4
```

```
/D/Garbage (master)
$ git checkout -b v1.0.4
Switched to a new branch 'v1.0.4'
```

```
/D/Garbage (v1.0.4)
$ git branch v1.0.3
```

②查看版本库中所有分支：

```
git branch -a
```

```
/D/Garbage (v1.0.4)
$ git branch -a
  master
  v1.0.3
* v1.0.4
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

③切换到某一分支：

```
git checkout v1.0.3
```

```
/D/Garbage (v1.0.4)
$ git checkout v1.0.3
Switched to branch 'v1.0.3'

/D/Garbage (v1.0.3)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
```

④删除某一分支：

```
git branch -D v1.0.4
```

⑤合并分支

```
 git merge v1.0.3
```

```
/D/Garbage (master)
$ git merge v1.0.3
Already up-to-date.
```

4.本地仓库与远程仓库同步问题

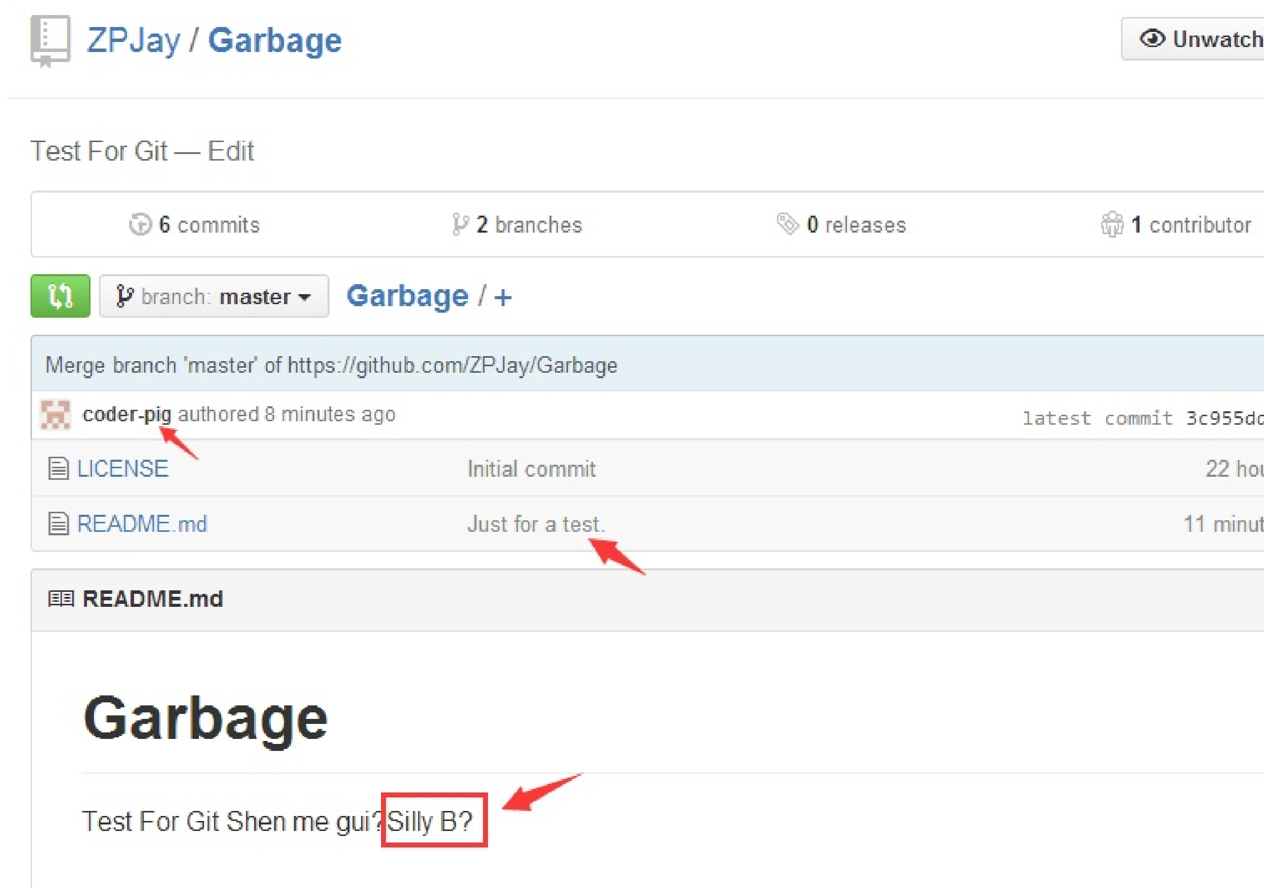
前面执行的这些分支操作都是在本地进行的，说了项目托管到GitHub上，我们肯定要跟远程仓库有交流是吧！ 我们去年前面已经试过用clone命令把项目下载到本地，那么我们修改后如何把代码同步到Github上呢？我们先对我们的本地仓库做一点点修改，接着git add和git commit本地准备后，然后：

```
git push origin master 或者直接 git push
```

将我们本地的内容提交上去：

```
/D/Garbage (master)
$ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/ZPJay/Garbage
   3ba3b8a..ed9e018  master     -> origin/master
 * [new branch]      v1.0.3     -> origin/v1.0.3
Already up-to-date!
Merge made by the 'recursive' strategy.
```

然后看下我们的Github,可以看到内容已经发生改变，而且提交者是我的另一个账号！



有同步到服务器，肯定有服务器同步到本地是吧，很简单，就一个

```
git pull
```

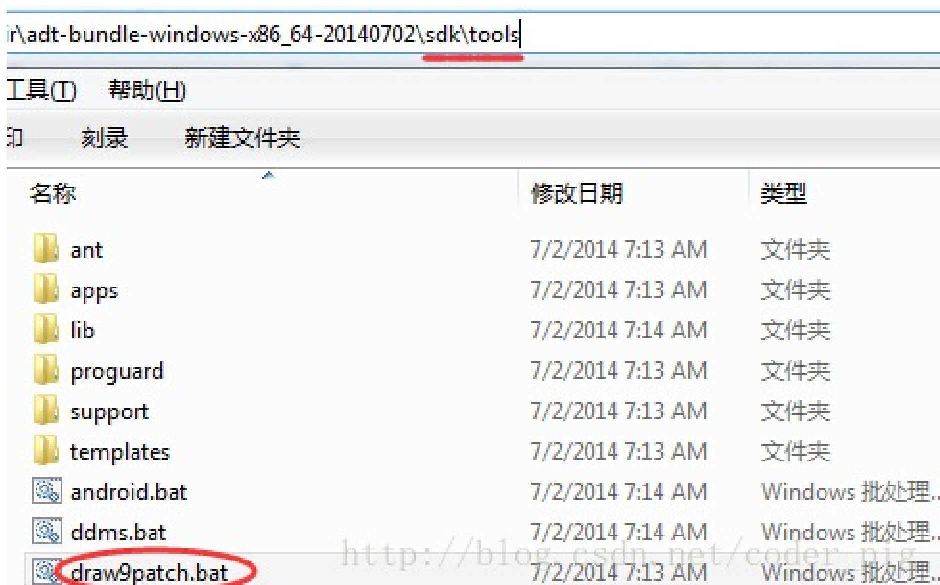
就可以

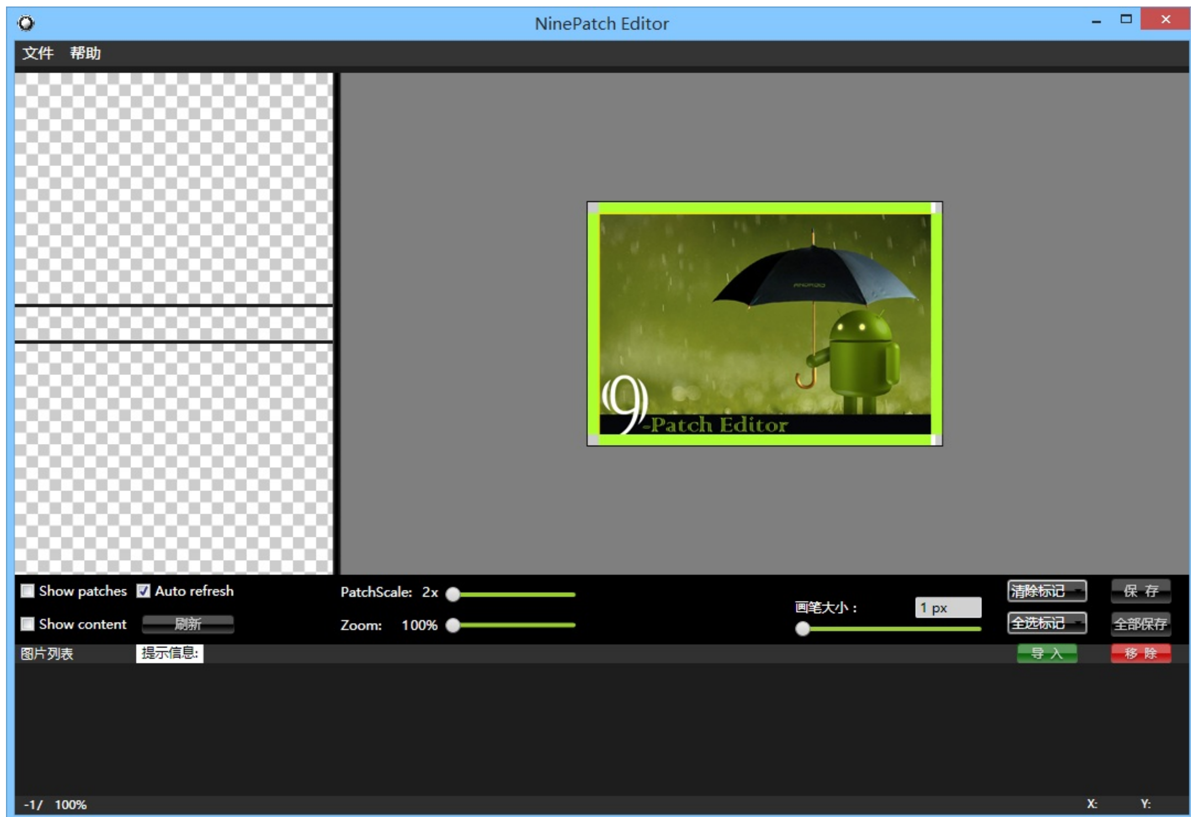
5.本节小结

好吧，本节就写那么多，相信你看到上面的Git教程还有一些冲突解决，分支管理，Bug分支等待高级的Git用法，考虑到这是入门教程，就不写那么深入了，有兴趣可以自己了解了解，说下自己公司目前的情况吧：①使用Github作为我们的项目管理工具：我们都是把项目托管到Github上的，然后有两个分支：开发和测试两个分支，每个版本一个分支，最后发布时才把分支合并到master上！提bug也是在上面提的，还是比较便利的！②使用Trello来做流程控制，也是比较简洁高效的！有兴趣的可以了解了解！另外，国内访问Github可能比较缓慢，而且如果是私有仓库是要收费的，如果公司没有使用代理或者是私人开发，可能略显鸡肋，不过可以考虑下使用国产的开源仓库：Git@OSC，由开源中国提供的，提供了1000个私人仓库，好像，感觉还不错，有兴趣的可以考虑将代码托管到这里：http://git.oschina.net/！就到这里，如果文中有错误纰漏，欢迎指出，谢谢~

1.6 .9(九妹)图片怎么玩

1.本节引言：

> 可能有的一些疑问：
>
> **1.**什么是**.9**图片？ 答：图片后缀名前有.9的图片,如pic1.9.png这样的图片
>
> **2. .9**图片能干嘛？ 答： 在图片拉伸的时候特定的区域不会发生图片失真，而不失真的区域可以由我们自己绘制 **3. .9**图片用什么做？ 答：工欲善其事，必先利其器，做.9图片的工具有：
>
> ①**Android SDK**自带：**draw9patch.bat**，不过这玩意出了好久，谷歌竟然没更新过...
>
> 
>
>
> ②**NinePatchEditor**，相比起自带的，做了一些优化，支持批量操作，而且界面看起来美观一点： 有兴趣的可以下载下，笔者平时用的这个，下载链接：NinePatchEditor.zip



③**NinePng**九图神器，手机版的.9处理工具，做得还是比较赞的，但是要连wifi互传图片，实际操作起来有点麻烦，功能还是比较强大的，有兴趣到相关应用市场搜索下载：

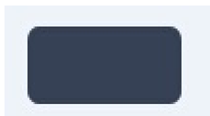


> ④**PhotoShop**，这就显得比较夸张了，一般用这个做.9图的都是美工，有兴趣的可搜下相关教程！

2. .9图片怎么做？

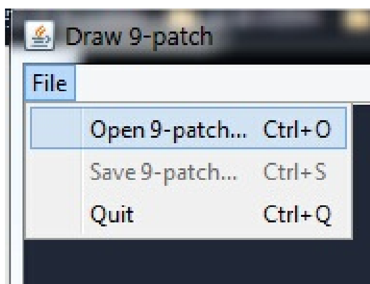
！！核心要点：左上拉伸，右下内容！！！！！！其实核心就上面的内容！先来找个图片试试手！

1. draw9patch.bat制作.9图实例：

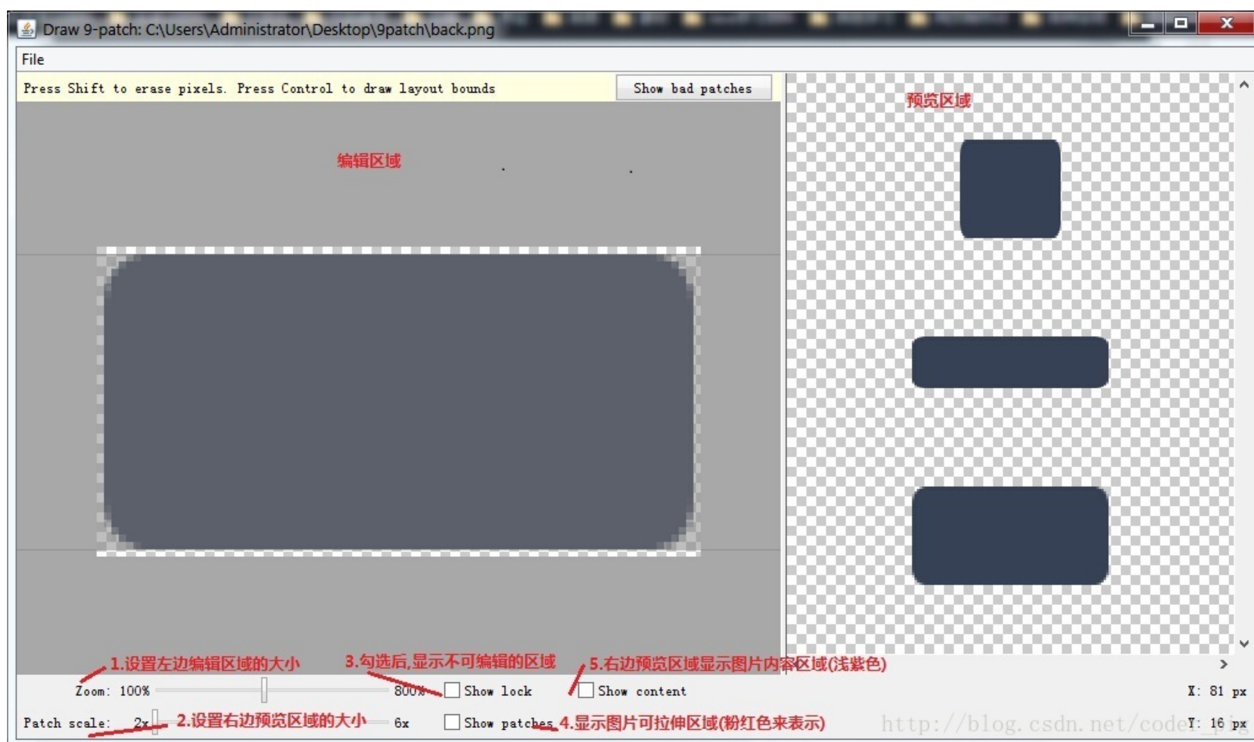
有这样的图片：，我们通过TextView的android:blackground可以设置为TextView的一个背景，内容少的时候还正常，一多起来就可能出现下面这种情况：会发现图片被拉伸变形了,很明显,这不合我们的需求,于是乎我们需要对这个图片来进行一些处理,让圆角部分的不随长度拉伸,中间部分才拉伸



打开我们的draw9patch.bat，点击左上角File，来到对应目录打开我们要处理的图片素材，接下来就可以看到我们工具的主界面了:

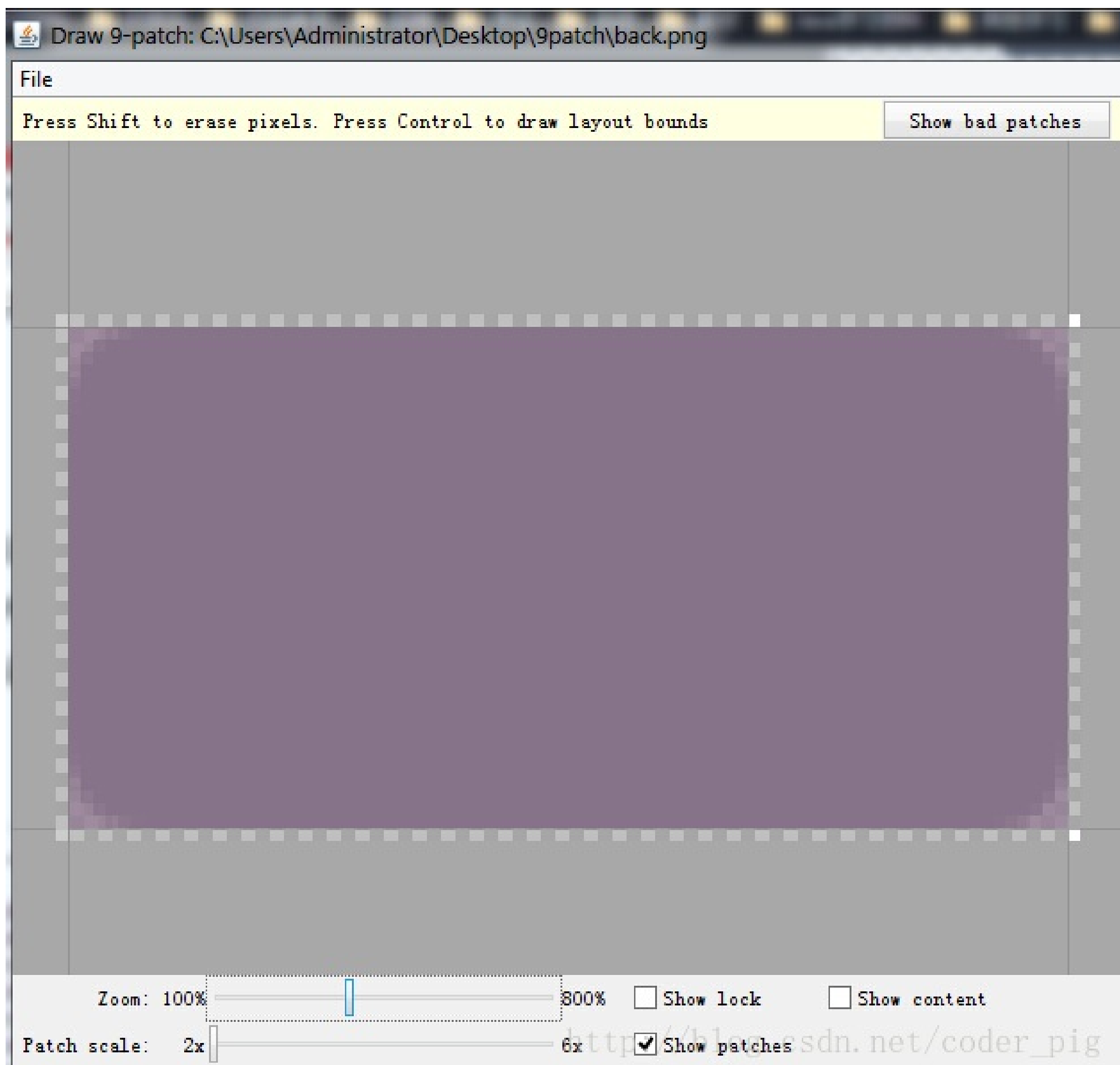


右面的预览区域分别是：纵向拉伸，横向拉伸，横纵都拉伸的预览

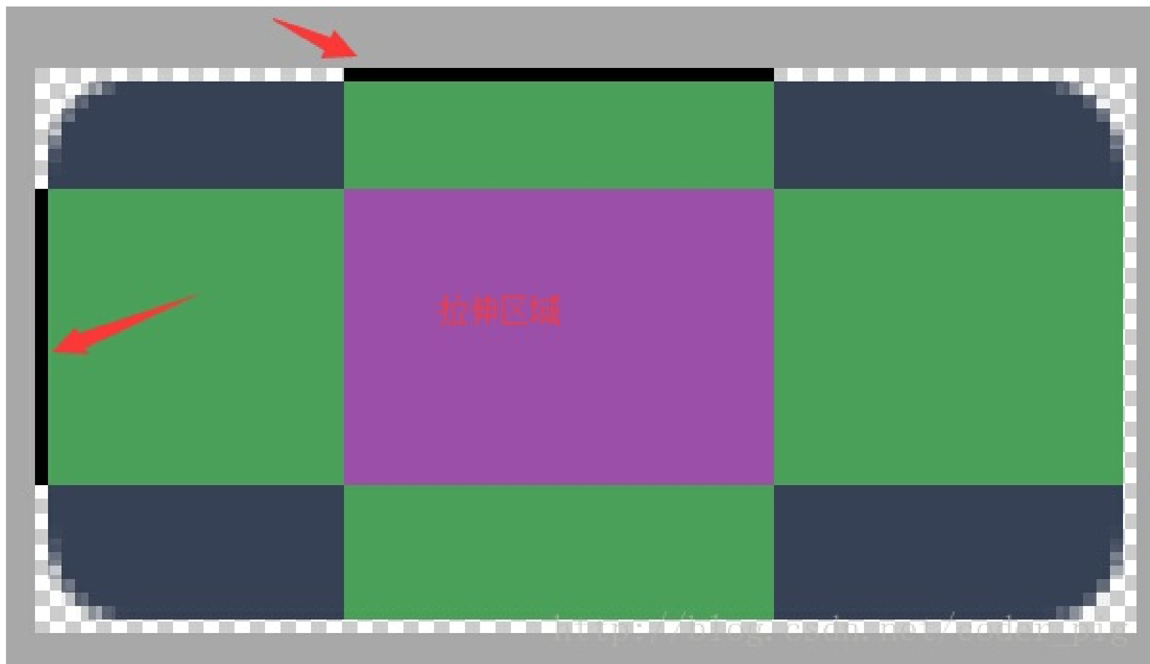


好的，接下来开始处理图片了：

Step 1.调Zoom和Patch scale:设置自己适合的缩放比例,勾选show patch 可以让Zoom足够大,因为后面我们需要处理"斑马线"



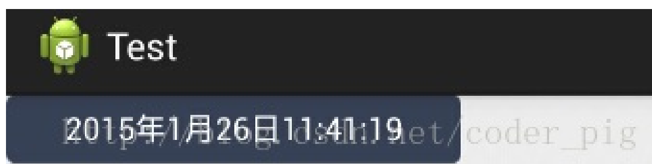
Step 2.接下来我们只需要在"斑马线"上进行操作就可以了: PS:黑色那条线是一条条点出来的,如果想消除点的话:按住shift点即可!



Step 3.保存图片,以.9.png结尾 比如这里保存的文件名是back.9.png



；嘿嘿，然后把他加入我们的工程，设置为TextView的背景：

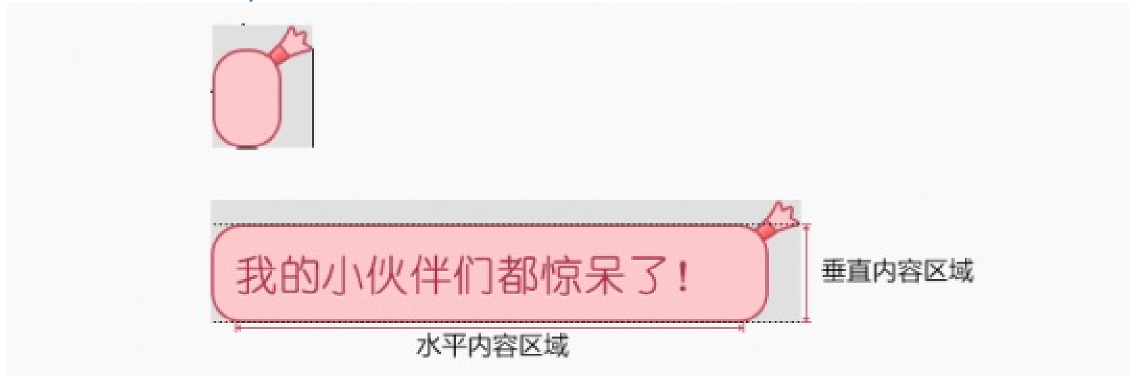


效果杠杠滴，接下来无论我们的显示的字符多长，都是图中这个结果，新技能get~

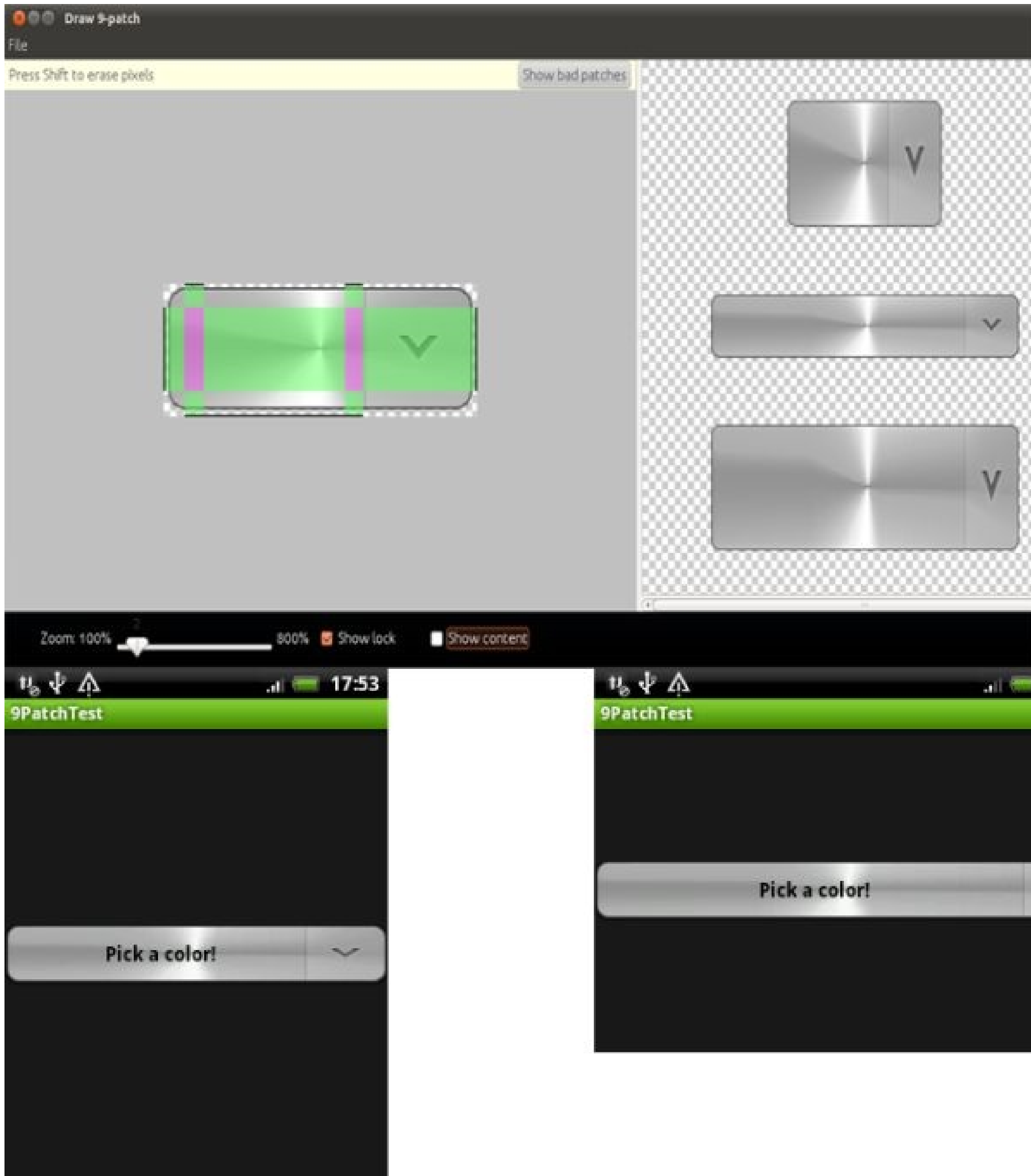
2.看下别人如何做.9图：

根据不同的情况我们可能需要做不同的.9图，下面欣赏下几个别人弄好的稍微复杂点的.9图的例子！ 例子：

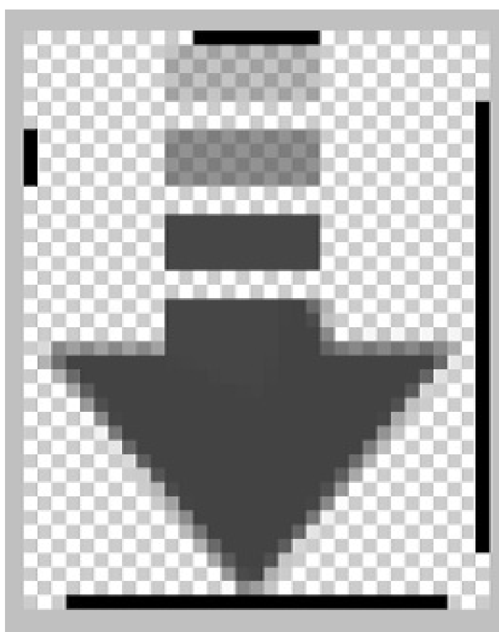
1.原文链接：http://www.miued.com/2074/好吧，这素材我喜欢，可以没有QAQ！



2.原文链接：http://blog.csdn.net/lizzy115/article/details/7950959



3.原文链接：http://www.cnblogs.com/vanezkw/archive/2012/07/19/2599092.html



3.本节小结：

好的，本节关于.9制作可拉伸图片的教程就到这里，还是比较简单的，记住我们的口诀：左上拉伸，右下内容！ 做几个.9图后相信你就深有体会了，再见~

1.7 界面原型设计

本节引言：

引用锤子科技视觉设计总监——罗子雄在重庆TEDx活动上说的一小段话：

> 每当我们看到一些美妙的设计的时候，很多人心里面会有一种冲动，这种冲动会让你们想去创造一些 新的东西，创造一些美妙的事物。

我们常说用户体验用户体验，用户使用你的软件，第一个会接触的是什么？没错，图形化界面(GUI)，简称UI，对于用户而言，最直观，给用户留下第一印像的是往往是程序的界面，而非功能！人，总喜欢美的东西，对吧？ 假如一样的功能，决定用户取向的，往往是UI！精美的UI！当然还有一些贴心的人性化操作等！ 下图两个计算机的App界面：


Genymotion for personal use - HTC Evo - 4.2.2 - API ...
GridLayoutDemo
0
回退
清空
+
1
2
3
-
4
5
6
*
7
8
9
/
.
0
=
http://blog.csdn.net/coder_pig


1024
大写转换
历史记录
C
DEL
÷
×
7
8
9
−
4
5
6
+
1
2
3
=
0
.
标准
科学
工具箱

我们先不说功能，从UI上讲，你喜欢哪个？由此，一个产品的UI是非常重要的，而产品的界面原型设计一般是由公司的产品经理+美工来完成的，需求分析 ->界面原型设计 ->我们来写代码！可能你觉得界面原型对我们而言并不没什么作用，但假如你以后想自己开发App呢？又或者你升做产品经理呢？嘿嘿！世事无绝对，以后的事，谁知道呢？公司的话，大部分使用的都是Axure Rp，但是这个东西比较难用！除了这个以外还有其他很多的原型设计工具：

> - Pencil
> - Framer
> - Shireframe
> - UIDesigner
> - Balsamiq Mockups
> - Mockup Builder
> - Mockup
> - FrameBox
> - iPhone Mockup
> - GOOFLOW
> - WireframeSketcher
> - FluidIA
> - Indigo Studio
> - Origami
> - Quartz Composer
> - Justproto
> - Avocado
> - PaintCode
> - Mockplus(摩客)
> - 墨刀等....

笔者使用过的是两个国产的界面原型设计工具，他们分别是：Mockplus(摩客)和墨刀 本文会简要的介绍下Mockplus的用法！

Mockplus原型工具的使用：

有网页版以及客户端版供你选择：Mockplus官网

Step 1：注册一个你自己的账号，然后新建文件进入编辑界面！(时间关系，这里笔者直接试用) 然后弹出一个原型风格的对话框给我们选择：素描跟线条！

提示 ×

点击下面的图片，可以切换到对应风格



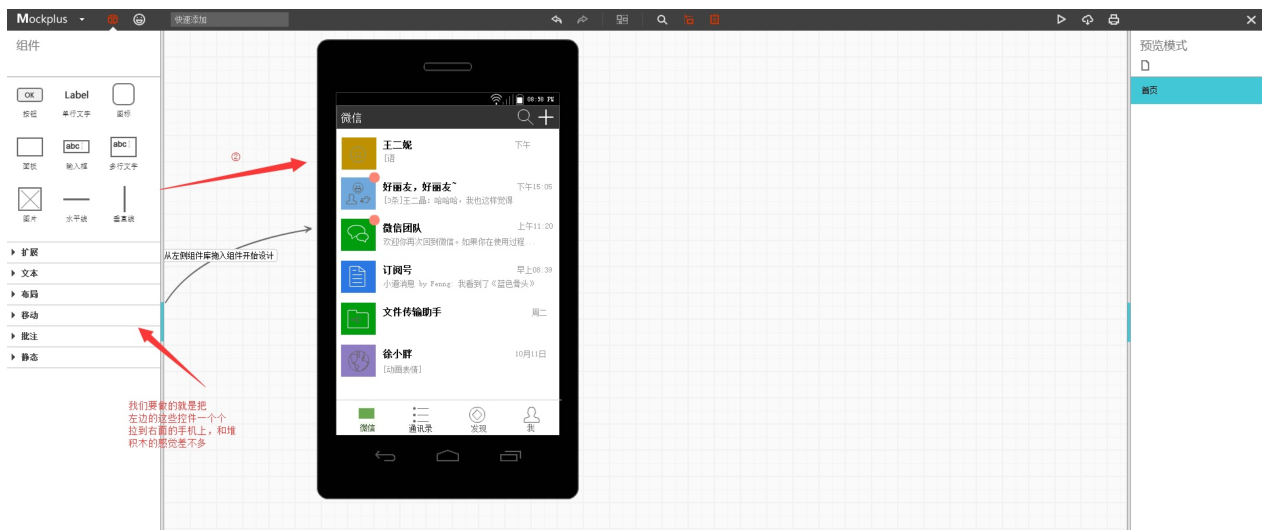
素描风格



线条风格

确定

笔者选择线条：



我们要做的就是从左边的组件栏中啦控件到手机界面上，当然，我们可以双击某个控件，自定义我们的样式，比如颜色，背景图片等！

Step 2：交互实现：界面原型除了界面外，肯定是少不了交互的，这里我们做一个简单的新闻应用的交互例子给大家体验下：我们在其中一个新闻类别中添加跳转链接，

接着我们可以点击右上角的播放按钮：



接下来我们就可以看具体的交互了，PS:这里可能是网页版的问题，有些图片显示



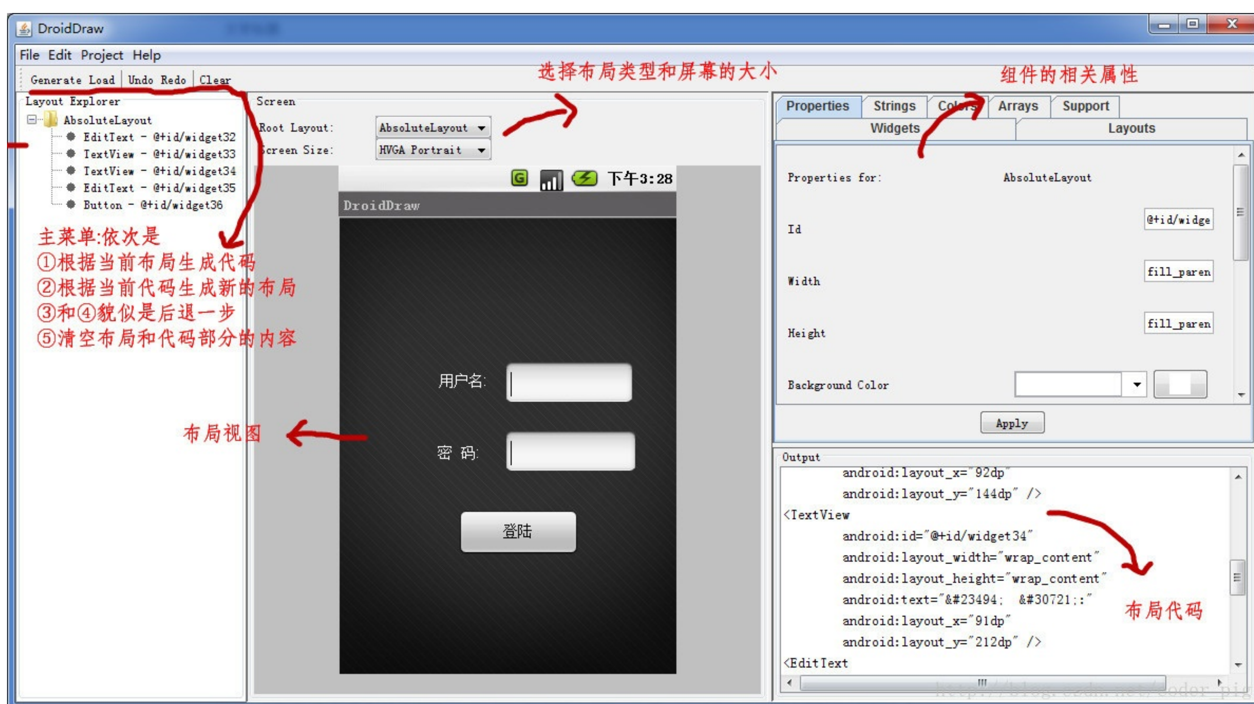
不出来！

好了，大概的用法就这些，大家可以自己摸索摸索，另外，如果生成文档是要升级为付费用户的哦！一个月6元不贵，就一个早餐钱，当然如果是自己玩玩而已就没必要开了！

Android自带DroidDraw工具设计Android界面：

其实Android也给我们提供了一个"老掉牙"的界面设计工具，跟上面这些高大上的界面原型工具相比，差几十条街，可以理解成一个分离的ADT，比ADT高级一点的功能就是自动生成代码...对于很少接触原型设计的朋友来说，花1，2分钟就可以掌握这个工具，还是比较值得！

工具界面：



图示已经写明相关的操作了,很简单,实践是检验真理的唯一标准,自己下来试试吧!!

软件下载：droiddrawr.jar

本节小结：

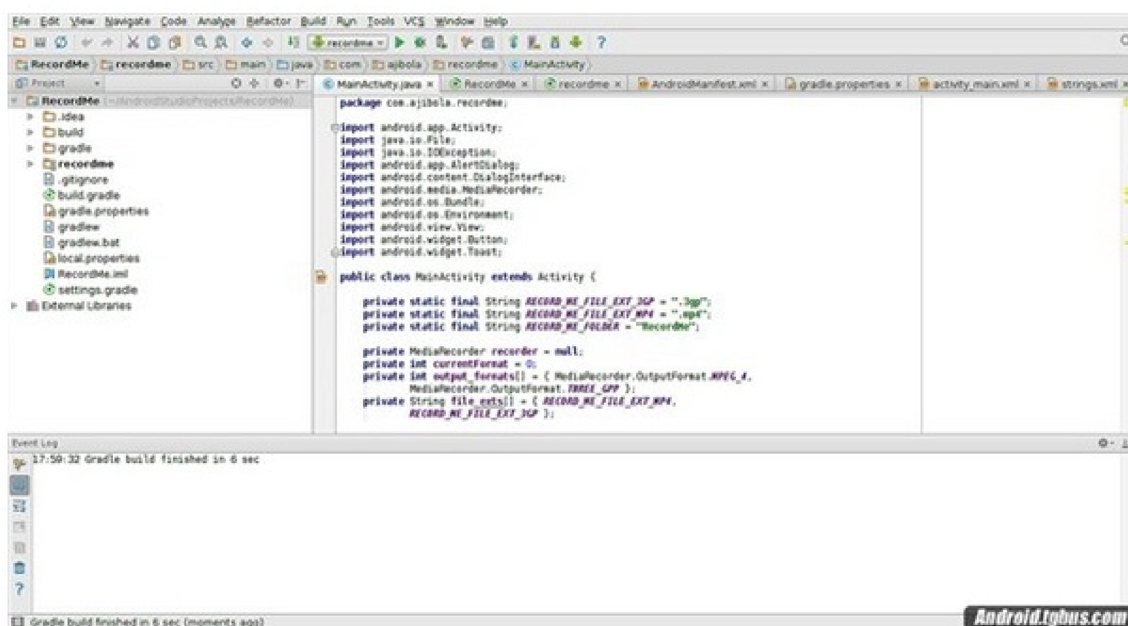
本节给大家介绍了下界面原型设计的概念，轻量级Mockplus(摩客)的简单实用，以及Android自带的DroidDraw工具，内容比较简单，还需大家自行实践理解！

1.8 工程相关解析(各种文件，资源访问)

本节引言：

前面讲了一堆看似和我们Android开发无关的东西是吧，当然是现在看似而已，以后你回头看就知道了！ 好吧，本节我们就来以前面创建的Hello World项目为入口，来了解工程结构， 以及Android中的资源访问的两种方式！后续教程使用的IDE是Android Studio，因为在前几天谷歌正式宣布，在年底前终止对其他IDE开发环境的支持！

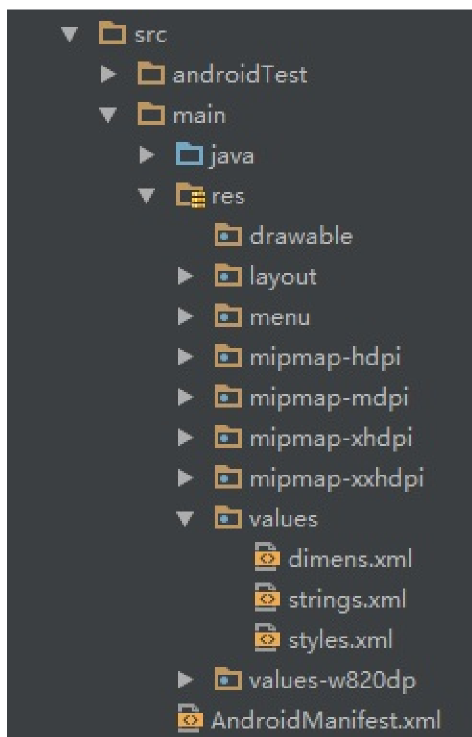
【巴士数码】谷歌在2013年为开发者提供的IDE环境工具Android Studio现在已经非常强大，因此谷歌现在又要关闭其他类似项目了。**谷歌宣布将在年底前中止对其他IDE开发环境的支持—开发者是时候正式向Eclipse说再见。**安卓产品经理Jamal Eason在声明中写道“谷歌将会全力专注于Android Studio编译工具的开发和技术支持，中止为Eclipse提供官方支持。包括中止对Eclipse ADT插件以及Android Ant编译系统的支持。”



Android Studio在两年前的Google I/O大会上首度曝光，不过一直到去年底才推出1.0正式版。这款IDE以常见的 Java 整合开发环境IntelliJ IDEA作为基础，能整合Google云端平台，并支援多种Android载具的Apps开发。之后的新版本还会包含C++/C语言编译的支持，同时在IDE中编写JAVA语言和C语言以及开展除错调试。

1.工程项目结构解析：

我们开发大部分时间都花在下面这个部分上：



接下来我们对关键部分进行讲解：

> - **java**：我们写Java代码的地方，业务功能都在这里实现
> - **res**：存放我们各种资源文件的地方，有图片，字符串，动画，音频等，还有各种形式的XML文件

1.res资源文件夹介绍：

PS：说到这个**res**目录，另外还有提下这个**assets**目录，虽然这里没有，但是我们可以自己创建，两者的区别在于是否前者下所有的资源文件都会在R.java文件下生成对应的资源id，而后者并不会；前者我们可以直接通过资源id访问到对应的资源；而后者则需要我们通过AssetManager以二进制流的形式来读取！对了，这个R文件可以理解为字典，res下每个资源都都会在这里生成一个唯一的id！

接着说下res这个资源目录下的相关目录：

PS:下述mipmap的目录，在Eclipse并不存在这个，Eclipse中都是drawable开头的，其实区别不大，只是使用mipmap会在图片缩放在提供一定的性能优化，分辨率不同系统会根据屏幕分辨率来选择hdpi，mdpi，xmdpi，xxhdpi下的对应图片，所以你解压别人的apk可以看到上述目录同一名称的图片，在四个文件夹下都有，只是大小和像素不一样而已！当然,这也不是绝对的,比如我们把所有的图片都丢在了drawable-hdpi下的话,即使手机 本该加载ldpi文件夹下的图片资源,但是ldpi下没有,那么加载的还会是hdpi下的图片! 另外,还有一种情况:比如是hdpi,mdpi目录下有,ldpi下没有,那么会加载mdpi中的资源! 原则是使用最接近的密度级别!另外如果你想禁止Android不跟随屏幕密度加载不同文件夹的资源,只需在AndroidManifest.xml文件中添加android:anyDensity="false"字段即可!

1.先说下图片资源：

> - **drawable**：存放各种位图文件，(.png，.jpg，.9png，.gif等)除此之外可能是一些其他的drawable类型的XML文件
> - **mipmap-hdpi**：高分辨率，一般我们把图片丢这里
> - **mipmap-mdpi**：中等分辨率，很少，除非兼容的的手机很旧
> - **mipmap-xhdpi**：超高分辨率，手机屏幕材质越来越好，以后估计会慢慢往这里过渡
> - **mipmap-xxhdpi**：超超高分辨率，这个在高端机上有所体现

2.接着说下布局资源：

> - layout：该目录下存放的就是我们的布局文件，另外在一些特定的机型上，我们做屏幕适配，比如480*320这样的手机，我们会另外创建一套布局，就行：layout-480x320这样的文件夹！

3.接下来说下菜单资源：

> - **menu**：在以前有物理菜单按钮，即menu键的手机上，用的较多，现在用的并不多，菜单项相关的资源xml可在这里编写，不知道谷歌会不会出新的东西来替代菜单了~

4.接下来说下values目录：

> - demens.xml：定义尺寸资源
> - string.xml：定义字符串资源
> - styles.xml：定义样式资源
> - colors.xml：定义颜色资源
> - arrays.xml：定义数组资源
> - attrs.xml：自定义控件时用的较多，自定义控件的属性！
> - theme主题文件，和styles很相似，但是会对整个应用中的Actvitiy或指定Activity起作用，一般是改变窗口外观的！可在Java代码中通过setTheme使用，或者在Androidmanifest.xml中为<application...>添加theme的属性！ PS:你可能看到过这样的values目录：values-w820dp，values-v11等，前者w代表平板设备，820dp代表屏幕宽度；而v11这样代表在API(11)，即android 3.0后才会用到的！

5.在接着说下这个raw目录： 用于存放各种原生资源(音频，视频，一些XML文件等)，我们可以通过openRawResource(int id)来获得资源的二进制流！其实和Assets差不多，不过这里面的资源会在R文件那里生成一个资源id而已

6.最后还有个动画的，动画有两种：属性动画和补间动画：

> - animator：存放属性动画的XML文件
> - anim：存放补间动画的XML文件

2.如何去使用这些资源

嗯，知道有什么资源，接下来就来了解该怎么用了： 前面也说了，我们所有的资源文件都会在R.java文件下生成一个资源id，我们可以通过这个资源id来完成资源的访问，使用情况有两种：Java代码中使用和XML代码中使用： **Java**代码中使用：

```
Java 文字：txtName.setText(getResources().getText(R.string.name));
图片：imgIcon.setBackgroundDrawableResource(R.drawable.icon);
颜色：txtName.setTextColor(getResouces().getColor(R.color.red));
布局：setContentView(R.layout.main);
控件：txtName = (TextView)findViewById(R.id.txt_name);
```

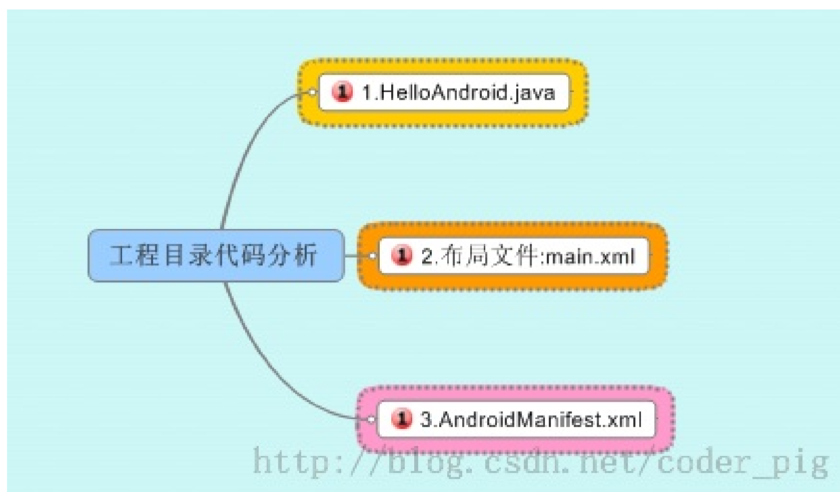
XML代码中使用：

通过@xxx即可得到，比如这里获取文本和图片:

```
<TextView android:text="@string/hello_world" android:layout_width='
```

2.深入了解三个文件：

好了，接下来我们就要剖析工程里三个比较重要的文件： MainActivity.java，布局文件：activity_main和Android配置文件：AndroidManifest.xml **PS**：图片内容可能有点差距，没时间做图，望体谅~



MainActivity.java：

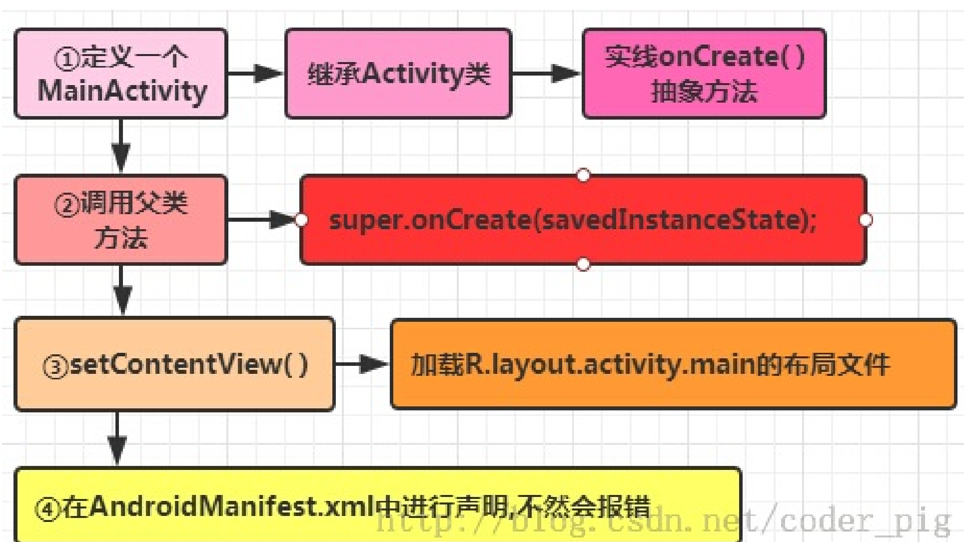
代码如下

```
package jay.com.example.firstapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

代码分析：



布局文件：activity_main.xml：

代码如下：

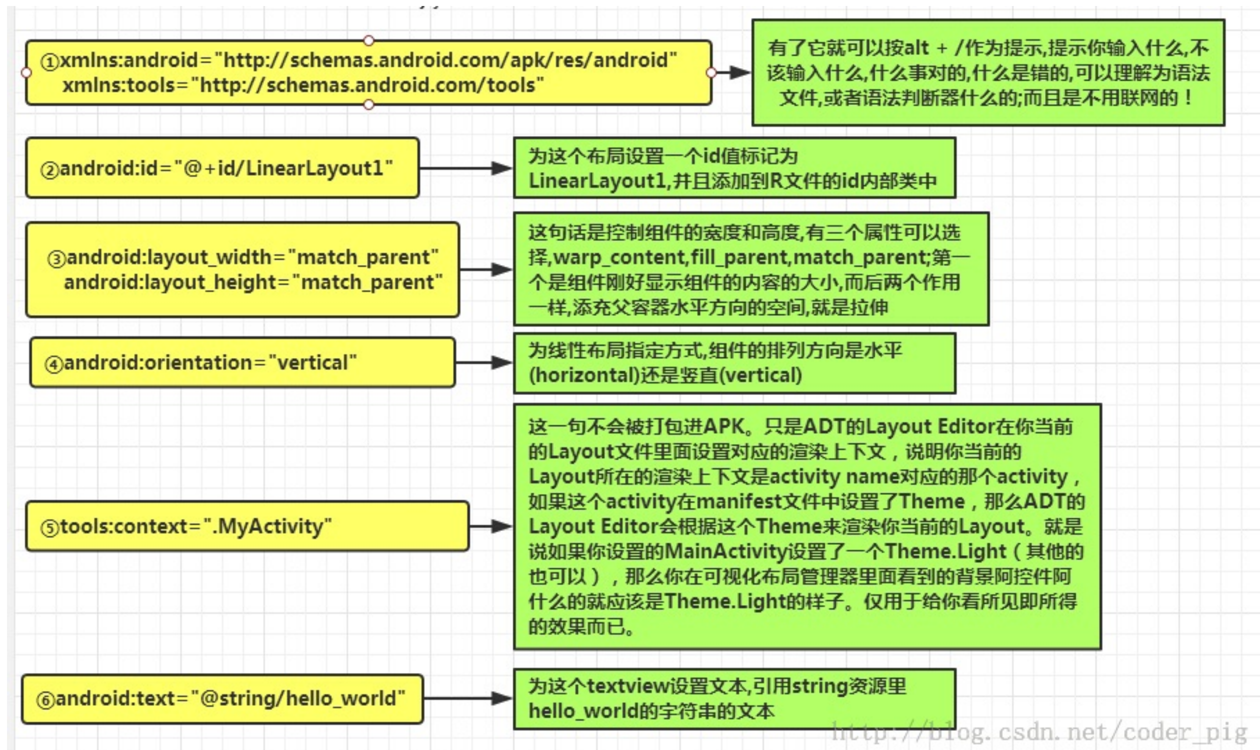
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

代码分析：

我们定义了一个LinearLayout线性布局，在xml命名空间中定义我们所需要使用的架构,来自于①



AndroidManifest.xml配置文件:

代码如下：

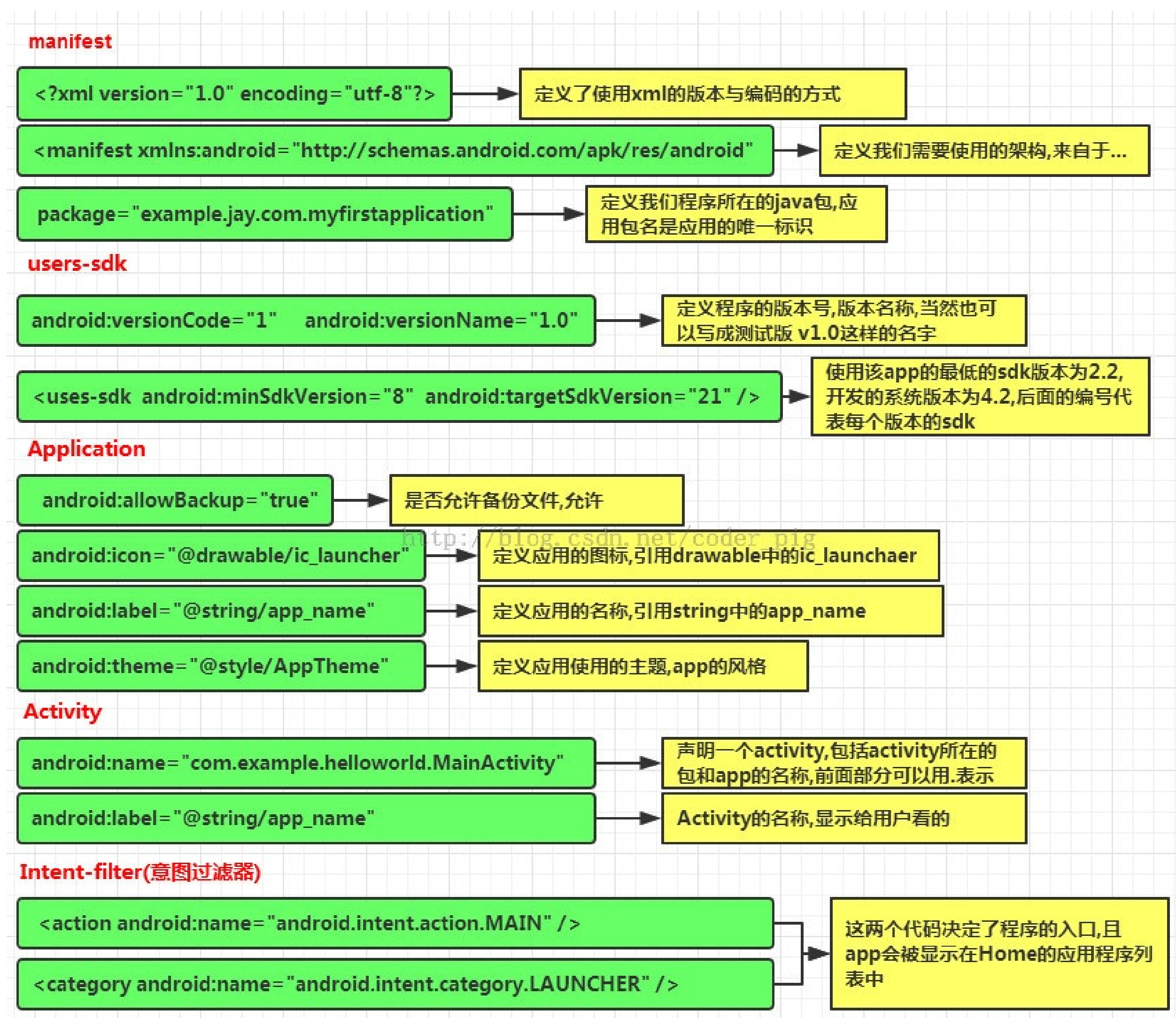
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androi
    package="jay.com.example.firstapp" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

代码分析：



除了上述内容外：

> ①如果app包含其他组件的话,都要使用类型说明语法在该文件中进行声明 Server:<server>元素 BroadcastReceiver<receiver>元素 ContentProvider<provider>元素 IntentFilter<intent-filter>元素</provider> </receiver></server>
>
> ②权限的声明: 在该文件中显式地声明程序需要的权限,防止app错误地使用服务, 不恰当地访问 资源,最终提高android app的健壮性 android.permission.SEND_SMS 有这句话表示app需要使用发送信息的权限,安装的时候就会提示用户, 相关权限可以在sdk参考手册查找！

本节小结：

本节我们对我们的Hello World项目进行了详细的了解，相关目录是做什么的，res下的资源文件有哪些，有什么作用，怎么使用这些资源！同时来通过画图的方式详细地讲解了工程中三个最重要的文件！到这里相信你对Android项目已经有个大体的了解了！谢谢~

1.9 Android程序签名打包

本节引言：

第一章的倒数第二节，本节给大家介绍的是如何将我们的程序打包成Apk文件，并且为我们的Apk签名！ 上一节中已经说了，我们后续的教程使用的IDE是Android Studio，所以本节讲解的也是AS(后面都这样 简称吧)下对项目进行打包签名！

1.什么是签名，有什么用：

> Android APP都需要我们用一个证书对应用进行数字签名，不然的话是无法安装到Android手机上的，平时我们调试运行时到手机上时，是AS会自动用默认的密钥和证书来进行签名；但是我们十几发布编译时，则不会自动签名，这个时候我们就需要进行手动签名了！ 为我们的APK签名有以下好处：
>
> - **1.**应用程序升级：如果你希望用户无缝升级到新的版本，那么你必须用同一个证书进行签名。这是由于只有以同一个证书签名，系统才会允许安装升级的应用程序。如果你采用了不同的证书，那么系统会要求你的应用程序采用不同的包名称，在这种情况下相当于安装了一个全新的应用程序。如果想升级应用程序，签名证书要相同，包名称要相同！
> - **2.**应用程序模块化： Android系统可以允许同一个证书签名的多个应用程序在一个进程里运行，系统实际把他们作为一个单个的应用程序，此时就可以把我们的应用程序以模块的方式进行部署，而用户可以独立的升级其中的一个模块。
> - **3.**代码或者数据共享： Android提供了基于签名的权限机制，那么一个应用程序就可以为另一个以相同证书签名的应用程序公开自己的功能。以同一个证书对多个应用程序进行签名，利用基于签名的权限检查，你就可以在应用程序间以安全的方式共享代码和数据了。 不同的应用程序之间，想共享数据，或者共享代码，那么要让他们运行在同一个进程中，而且要让他们用相同的证书签名。 ————上述内容摘自:android 为什么需要签名

2.Android Studio如何打包签名：

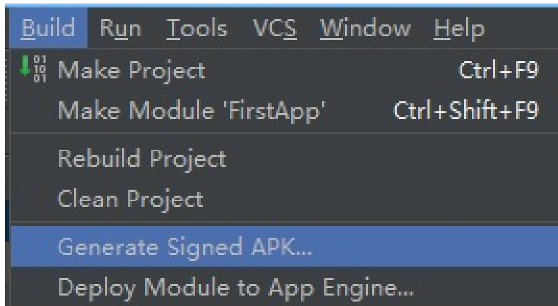
好的，因为学习本课程的都是初学者，多渠道打包的内容以后再进行讲解！本节只讲最简单的打包签名 对了，1中说的调试时默认生成的apk在：app/build/outputs/apk目录下！ 和Eclipse并不相同，Eclipse是在bin目录下生成的！

电脑 ▸ 文档 (D:) ▸ ASCode ▸ FirstApp ▸ app ▸ build ▸ outputs ▸ apk

名称	修改日期	类型	大小
app-debug.apk	2015/6/17 星期...	apk file	1,027 KB
app-debug-unaligned.apk	2015/6/17 星期...	apk file	1,027 KB

好的，打开我们的AS上的Hello World项目，点击菜单：

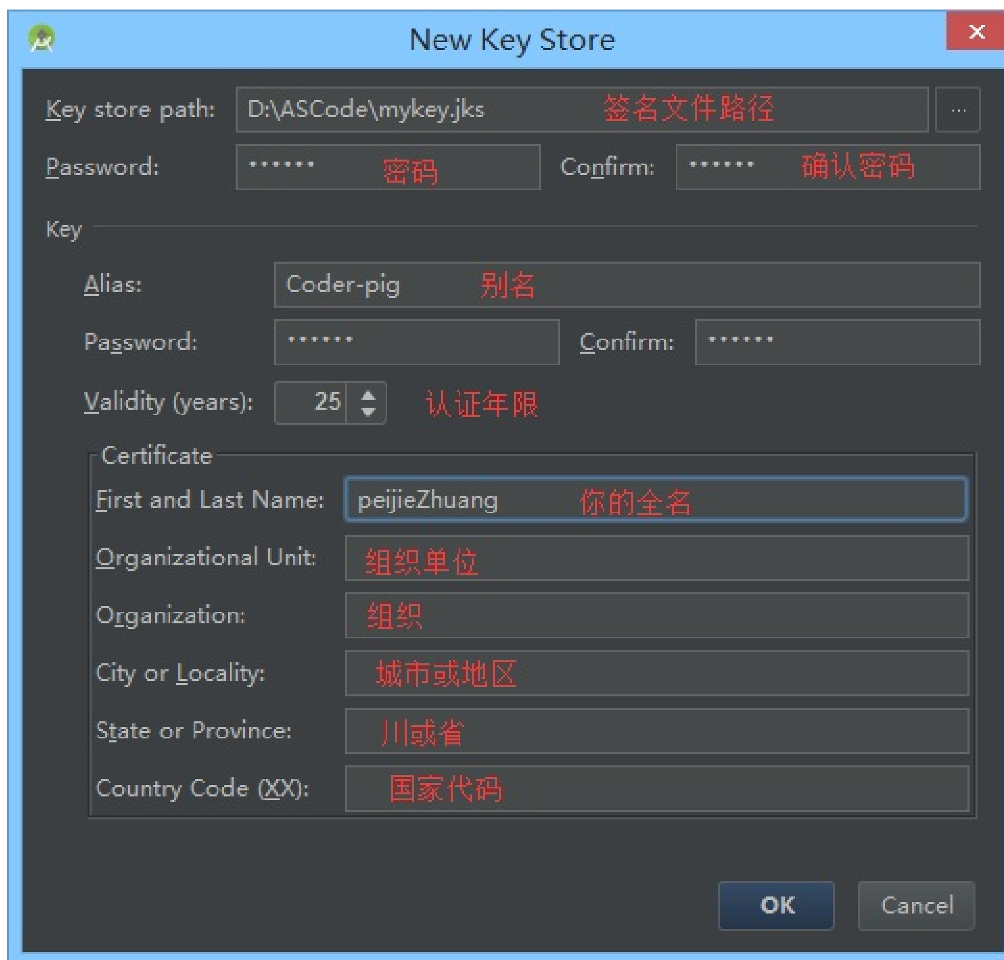
①Build -> Generate Signed APK...



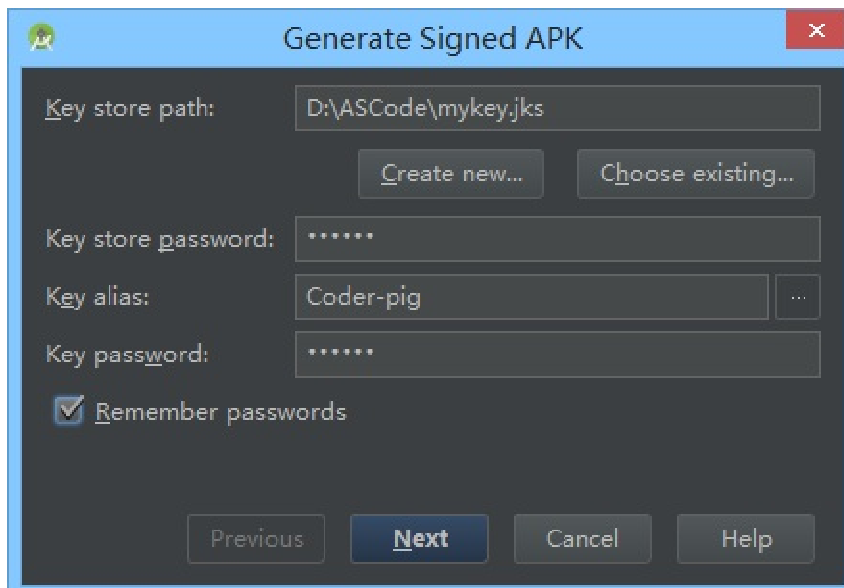
②弹出窗口，如果没有key，就创建一个，有的话就选择存在的Key



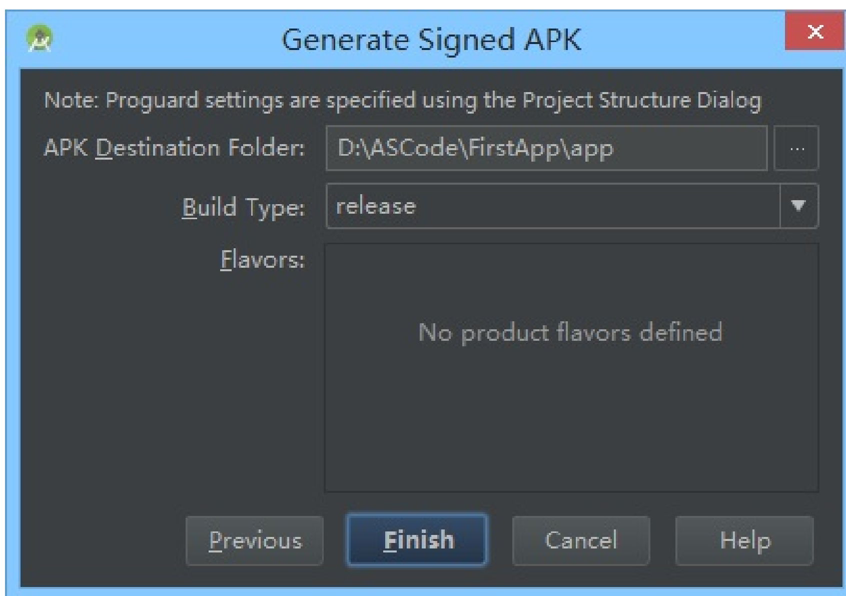
③没有，我们新建一个，可根据自己需要填写相关项：



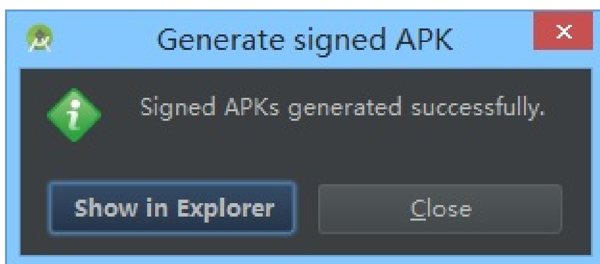
④好的，点击OK后，可以看到我们密码的信息，可能需要我们填入密码了，填写下：



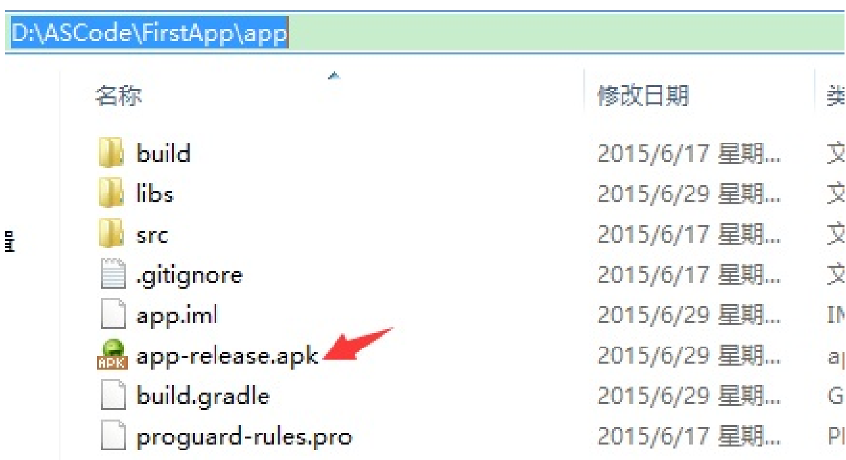
⑤点击Next：



⑥点击Finish稍等一会儿会出现下述提示，说明应用已经打包签名成功了：



⑦可以看到打包后的APK已经安详地躺在我们的app目录下了：



⑧到第七步就已经打包签名完成了，如果你要验证是否签名，只需要输入下述cmd指令

```
/D/ASCode/FirstApp/app
$ jarsigner -verbose -certs -verify app-release.apk
```

```
  s = 已验证签名
  m = 在清单中列出条目
  k = 在密钥库中至少找到了一个证书
  i = 在身份作用域内至少找到了一个证书

jar 已验证。
```

本节小结

打包Android APK的方法还有很多，命令行，或者Gradle，ANT，MAVEN等等，方法有很多，本节讲解最简单的通过图形化界面打包签名的方式！好了，本节就到这里，最简单的打包签名方法get了没？

1.11 反编译APK获取代码&资源

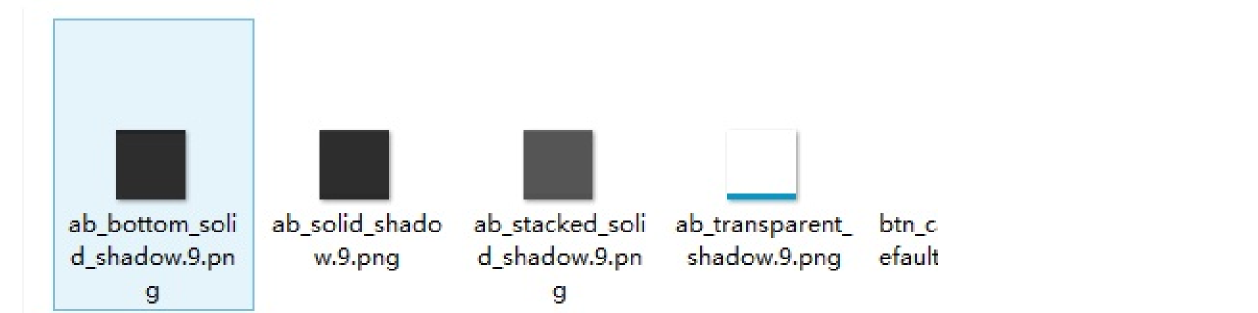
本节引言

> "反编译Apk"，看上去好像好像很高端的样子，其实不然，就是通过某些反编译软件，对我们的APK进行反编译，从而获取程序的源代码，图片，XML资源等文件；不知道你有没有这样做过，看到一个别人的一个APP界面做得很精美，或者你看上别人的图片素材，简单点的，我们可以下载别人的APK，然后改下后缀名，改成xxx.zip，然后解压：笔者随便解压了一个APK：

assets	2015/6/30 星期...	文件夹	
com	2015/6/30 星期...	文件夹	
lib	2015/6/30 星期...	文件夹	
META-INF	2015/6/30 星期...	文件夹	
org	2015/6/30 星期...	文件夹	
res	2015/6/30 星期...	文件夹	
actors.properties	2015/2/1 星期日 ...	PROPERTIES 文件	1 KB
AndroidManifest.xml	2015/2/1 星期日 ...	XML 文件	7 KB
classes.dex	2015/2/1 星期日 ...	DEX 文件	2,402 KB
library.properties	2015/2/1 星期日 ...	PROPERTIES 文件	1 KB
reflect.properties	2015/2/1 星期日 ...	PROPERTIES 文件	1 KB
resources.arsc	2015/2/1 星期日 ...	ARSC 文件	366 KB
rootdoc.txt	2015/2/1 星期日 ...	文本文档	2 KB

> 我们可以打开res目录，获取里面的图片素材

这台电脑 ▸ 下载 ▸ XunLuVPN1(www.greenxf.com) ▸ res ▸ drawable-hdpi-v4



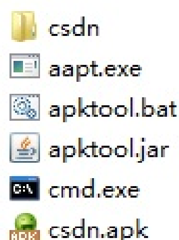
> 但是，这种方法，获得的只会是一些.png，或者.jpg这样的位图文件资源，如果是xml类的资源，打开我们会发现是乱码，并且假如我们想看APK程序的Java代码，也是行不通的，因为他们都打被打包到classes.dex文件中！但是反编译可以解决你的需要~另外，切勿拿反编译来做违法的事，比如把人家的APK重新打包后使用自己的签名然后发布到相关市场...另外，我们是参考别人的代码，而不是完全拷贝！！！切记！！

1.要准备的三个工具

> 1. **apktool**：获取资源文件，提取图片文件，布局文件，还有一些XML的资源文件
> 2. **dex2jar**：将APK反编译成Java源码(将classes.dex转化为jar文件)
> 3. **jd-gui**：查看2中转换后的jar文件，即查看Java文件 为了方便各位读者，这里将三个打包到一起放到云盘中，又需要的可以进行下载：反编译相关的三个工具.zip

2.使用apktool反编译APK获得图片与XML资源：

把下载好的apktool解压后，我们可以看到下述文件(忽略那两个csdn，一个是反编译的apk，一个是反编译后文件)：

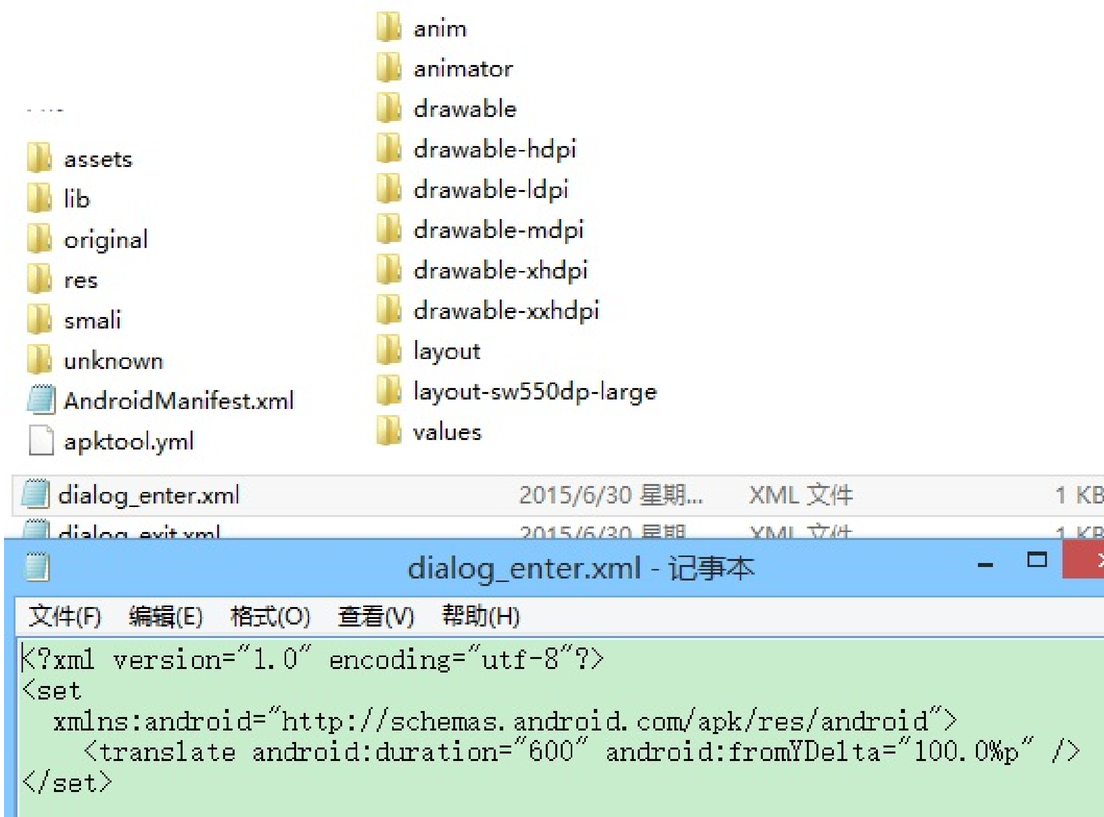


接下来，双击cmd.exe，来到命令行，键入：**apktool.bat d csdn.apk** 即可，Enter回车：

```
E:\Coding\Install\apktool\apktool2.0.0b9\apktool2.2>apktool.bat d csdn.apk
I: Using Apktool 2.0.0-Beta9 on csdn.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\dbj\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
W: Cant find 9patch chunk in file: "drawable-xhdpi/tag_add_icon.9.png". Renaming
 it to *.png.
I: Decoding values */* XMLs...
I: Baksmaling...
testI: Copying assets and libs...
I: Copying unknown files/dir...
I: Copying original files...

E:\Coding\Install\apktool\apktool2.0.0b9\apktool2.2>
```

然后就可以看到生成的csdn文件夹，里面就有我们想要资源



好的，就是XML资源到手了是吧！图片素材也到手了！

3.使用dex2jar将classes.dex转换成jar文件：

把下载好的dex2jar文件夹解压，apk解压后中的classes.dex复制到dex2jar.bat所在的目录下：

名称	修改日期	类型	大小
lib	2014/10/27 星期...	文件夹	
classes.dex　复制到这里	2015/5/29 星期...	DEX 文件	4,121 KB
d2j_invoke.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j_invoke.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-baksmali.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-baksmali.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-dex2jar.bat　看我	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-dex2jar.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-dex2smali.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-dex2smali.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-dex-recompute-checksum.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-dex-recompute-checksum.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-jar2dex.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-jar2dex.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-jar2jasmin.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-jar2jasmin.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-jasmin2jar.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-jasmin2jar.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-smali.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-smali.sh	2014/10/27 星期...	Shell Script	2 KB
d2j-std-apk.bat	2014/10/27 星期...	Windows 批处理...	1 KB
d2j-std-apk.sh	2014/10/27 星期...	Shell Script	2 KB

打开cmd，来到这个目录下：键入：**d2j-dex2jar.bat classes.dex**

```
E:\Coding\Install\apktool\dex2jar-2.0>d2j-dex2jar.bat classes.dex
dex2jar classes.dex -> .\classes-dex2jar.jar
```

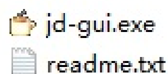
接着我们可以看到，生成了一个jar包：

名称	修改日期	类型	大小
lib	2014/10/27 星期...	文件夹	
classes.dex	2015/5/29 星期...	DEX 文件	4,121 KB
classes-dex2jar.jar	2015/6/30 星期...	Executable Jar File	4,125 KB

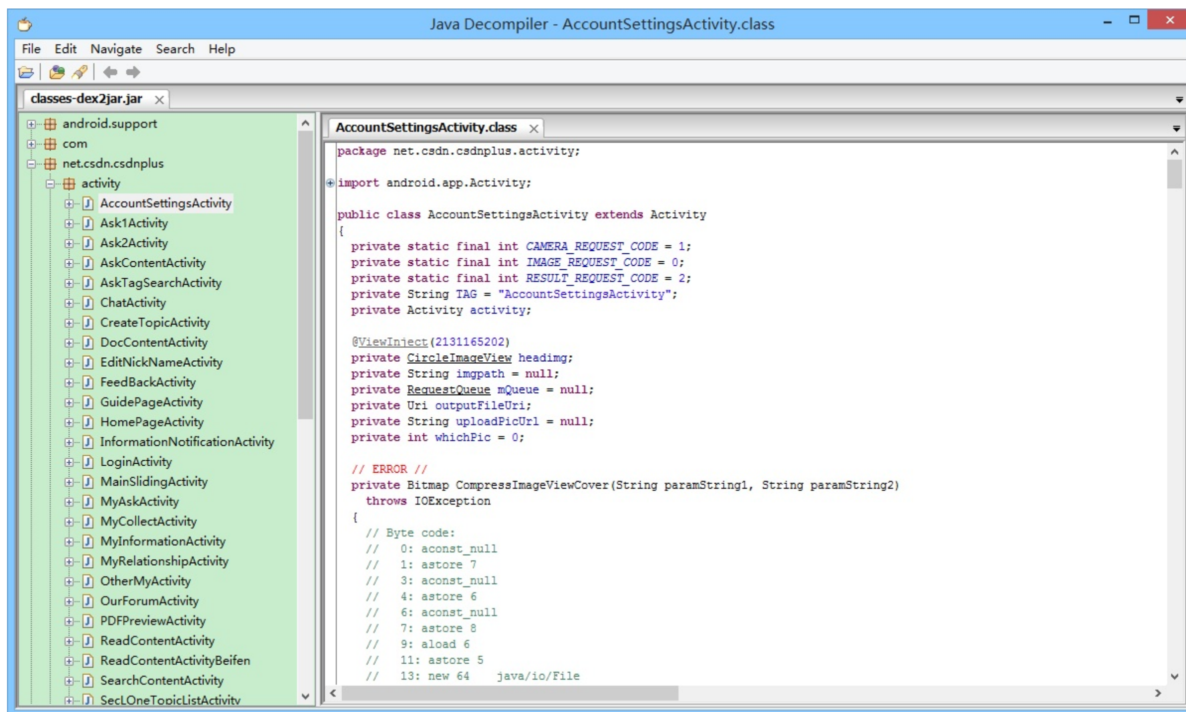
好的，转换完成！

4.使用jd-gui查看jar包中的Java代码：

好的，打开jd-gui的文件夹



打开后，打开我们3中转换后的jar包，我们可以看见里面的代码：



csdn的客户端竟然不混淆代码...可能是本着开源的精神吧，给我们学习代码吧！一般的话，apk发布都会 进行混淆，然后进行一些加密，或者使用第三方的加密平台，用的比较多的"爱加密"，有兴趣的也自行百度查看更加详细的介绍！

本节小结

好的，关于APK的反编译就介绍到这里，相信你已经摩拳擦掌想要试试了，那就试试吧，最后提醒一句，别做坏事！尊重别人的劳动成果！另外，关于第一大章环境搭建相关以及一些常用开发技巧就到这里，下一节开始我们就来进行本系列教程的第二章——Android中的常用UI控件的学习了！因相关的基本控件较多，估计有几十个，如果一直学控件可能没什么意思，可能并行写教程，每天学一个控件 + 一点其他的知识点这样，笔者要构思构思，敬请期待~谢谢~

2.1 View与ViewGroup的概念

本节引言

告别了第一章，迎来第二章——Android中的UI（User Interface）组件的详解， 而本节我们要学习的是所有控件的父类View和ViewGroup类！突发奇想，直接翻译官方文档对 这两个东西的介绍吧， 对了， 天朝原因，google上不去，Android developer上不去， 我们可以 改hosts或者用vpn代理， 当然也可以像笔者一样使用国内的API镜像，这里分享个吧：
http://androiddoc.qiniudn.com/guide/topics/ui/overview.html 这个镜像是5.0的API！

UI Overview

在Android APP中，所有的用户界面元素都是由View和ViewGroup的对象构成的。View是绘制在屏幕上的用户能与之交互的一个对象。而ViewGroup则是一个用于存放其他View（和ViewGroup）对象的布局容器！ Android为我们提供了一个View和ViewGroup子类的集合，集合中提供了一些常用的输入控件(比如按钮和文本域)和各种各样的布局模式（比如线性或相对布局）

User Interface Layout

你的APP的用户界面上的每一个组件都是使用View和ViewGroup对象的层次结构来构成的，比如 图 1。每个ViewGroup都是要给看不见的用于组织子View的容器，而它的子View可能是输入控件 或者在UI上绘制了某块区域的小部件。有了层次树，你就可以根据自己的需要，设计简单或者复 杂的布局了(布局越简单性能越好)

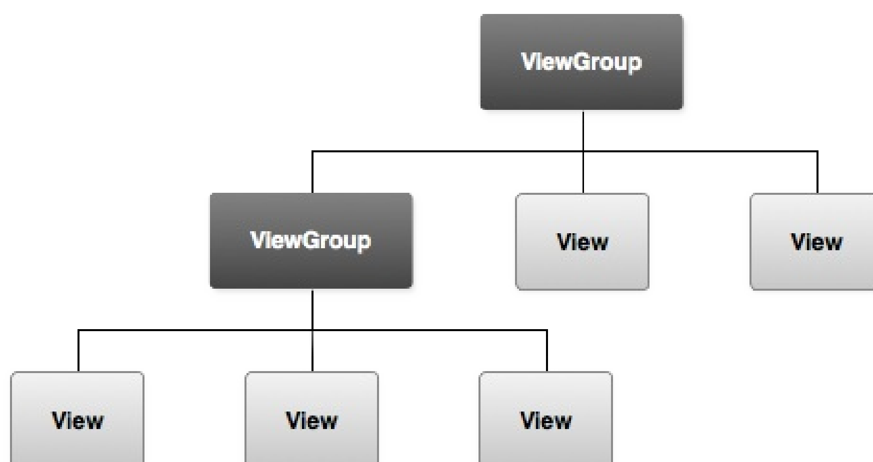


图 1.一个UI布局的层次结构的插图

定义你的布局，你可以在代码中实例化View对象并且开始构建你的树，但最容易和最高效的方式来定义你的布局则是使用一个XML文件，用XML来构成布局更加符合人的阅读习惯，而XML类似与HTML 使用XML元素的名称代表一个View。所以<

TextView >元素会在你的界面中创建一个TextView控件，而一个< LinearLayout >则会创建一个LinearLayout的容器！ 举个例子，一个简单简单的垂直布局上面有一个文本视图和一个按钮，就像下面这样：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button" />
</LinearLayout>
```

当你的App加载上述的布局资源的时候，Android会将布局中的每个节点进行实例化成一个个对象，然后你可以为这些定义一些额外的行为，查询对象的状态，或者修改布局。 完整创建UI布局的引导，请参考XML Layouts

User Interface Components

你无需全部用View和ViewGroup对象来创建你的UI布局。Android给我们提供了一些app控件，标准的UI布局，你只需要定义内容。这些UI组件都有其属性介绍的API文档，比如操作栏，对话框和状态通知栏等。

本节小结

好吧，翻译可能比较拗口，哎，英语盲尽力了，简单归纳下上述内容：

> Android里的图形界面都是由View和ViewGroup以及他们的子类构成的： **View**：所有可视化控件的父类,提供组件描绘和时间处理方法 **ViewGroup**：View类的子类，可以拥有子控件,可以看作是容器 Android UI中的控件都是按照这种层次树的结构堆叠得，而创建UI布局的方式有两种， 自己在Java里写代码或者通过XML定义布局，后者显得更加方便和容易理解！ 也是我们最常用的手段！另外我们一般很少直接用View和ViewGroup来写布局，更多的 时候使用它们的子类控件或容器来构建布局！

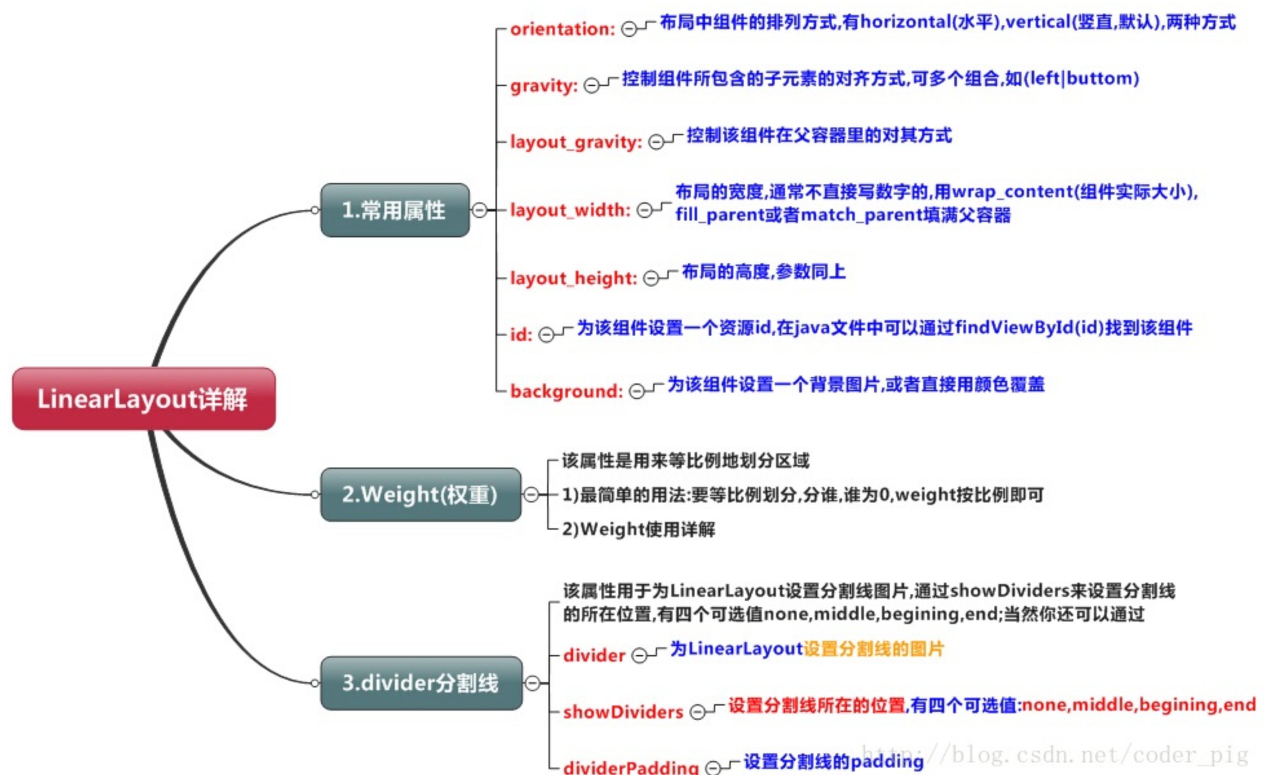
恩呢，对View和ViewGroup有个大概了解即可，平时我们是不会直接用的，一般是自定义View的时候 才会使用这两个东西！

2.2.1 LinearLayout(线性布局)

本节引言

本节开始讲Android中的布局，Android中有六大布局,分别是: LinearLayout(线性布局)，RelativeLayout(相对布局)，TableLayout(表格布局) FrameLayout(帧布局)，AbsoluteLayout(绝对布局)，GridLayout(网格布局) 而今天我们要讲解的就是第一个布局，LinearLayout(线性布局)，我们屏幕适配的使用 用的比较多的就是LinearLayout的weight(权重属性),在这一节里,我们会详细地解析 LinearLayout,包括一些基本的属性,Weight属性的使用,以及比例如何计算,另外还 会说下一个用的比较少的属性:android:divider绘制下划线！

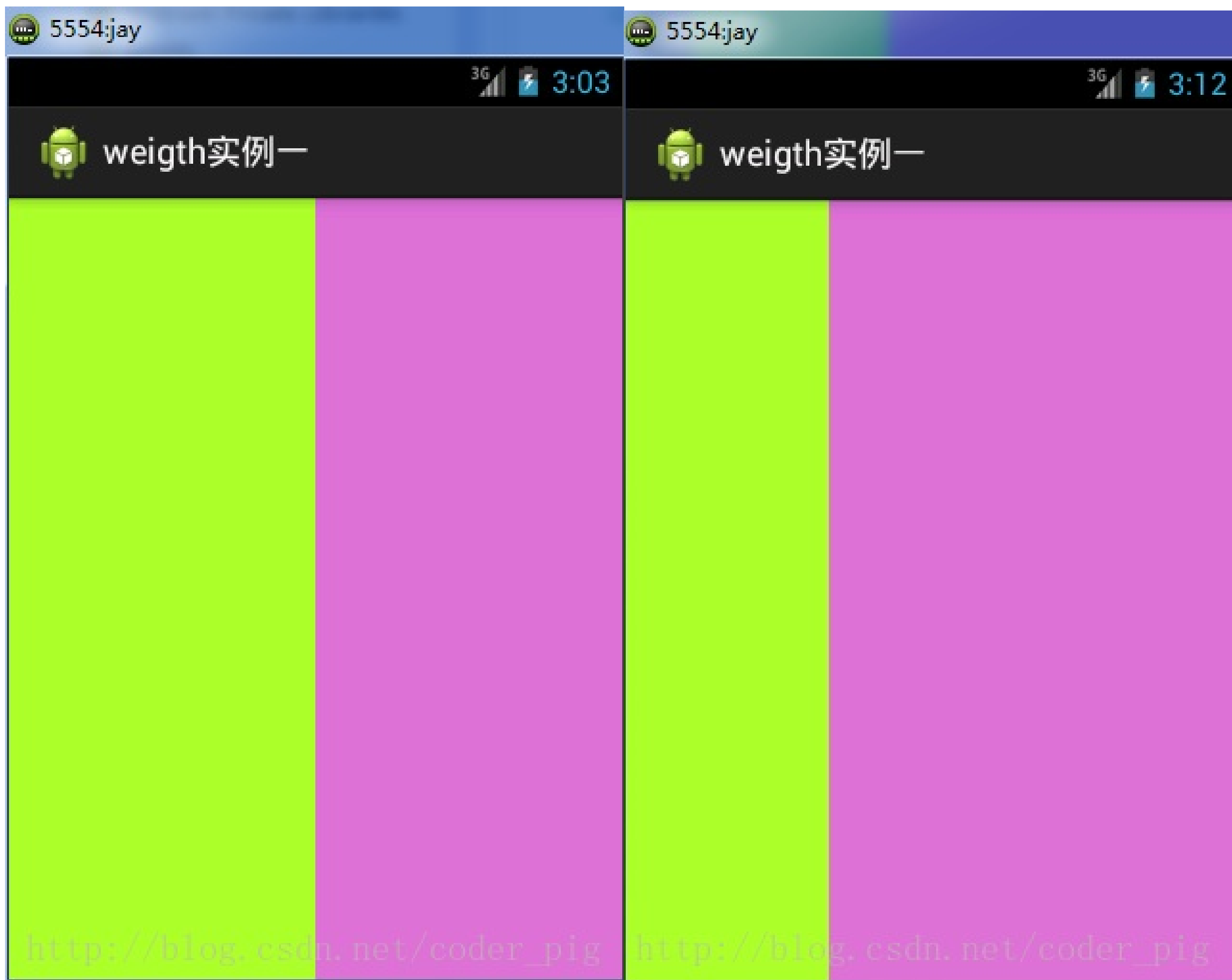
1.本节学习图



2.weight(权重)属性详解：

①最简单用法：

如图：



实现代码：

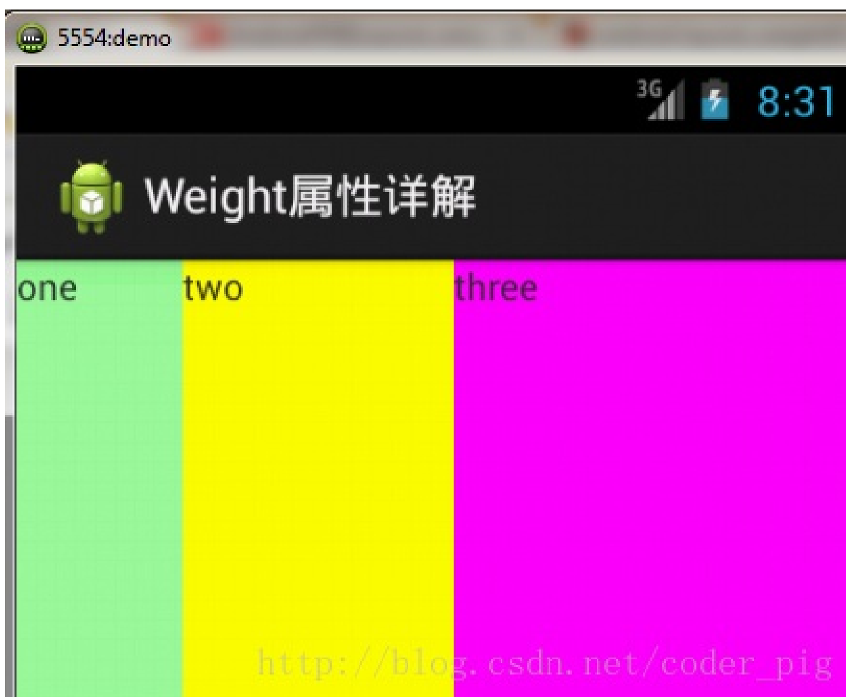
```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

要实现第一个的1:1的效果,只需要分别把两个LinearLayout的weight改成1和1就可以了 用法归纳: 按比例划分水平方向:将涉及到的View的android:width属性设置为0dp,然后设置为android weight属性设置比例即可;类推,竖直方向,只需设android:height为0dp,然后设weight属性即可！ 大家可以自己写个竖直方向的等比例划分的体验下简单用法！

②weight属性详解:

当然,如果我们不适用上述那种设置为0dp的方式,直接用wrap_content和match_parent的话, 则要接着解析weight属性了,分为两种情况,wrap_content与match_parent！另外还要看 LinearLayout的orientation是水平还是竖直,这个决定哪个方向等比例划分

1)wrap_content比较简单,直接就按比例的了



实现代码：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

2)match_parent(fill_parent):这个则需要计算了

我们写这段简单的代码：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

运行效果图：



> 这个时候就会有疑问了,怎么会这样,这比例是2:1吧,那么three去哪了?代码里面明明有 three的啊,还设置了3的,而1和2的比例也不对耶,1:2:3却变成了2:1:0,怎么会这样呢? 答:这里其实没那么简单的,还是需要我们计算的,网上给出的算法有几种,这里就给出笔者 觉得比较容易理解的一种: **step 1**:个个都是fill_parent,但是屏幕只有一个啦,那么1 - 3 = - 2 fill_parent **step 2**:依次比例是1/6,2/6,3/6 **step 3**:先到先得,先分给one,计算: 1 - 2 *(1/6) = 2/3 fill_parent* 接着到*two,*计算*: 1 - 2* (2/6) = 1/3 fill_parent 最后到three,计算 1 - 2 *(3/6) = 0 fill_parent *step 4*:所以最后的结果是:one占了两份,two占了一份,three什么都木有 以上就是为什么three没有出现的原因了,或许大家看完还是有点蒙,没事,我们举多几个例子试试就知道了!

比例为:**1:1:1**



按照上面的计算方法算一次,结果是:1/3 1/3 1/3,没错

接着我们再试下:**2:3:4**



计算结果:5/9 3/9 1/9,对比效果图,5:3:1,也没错,所以这个计算方法你可得mark下了!

③Java代码中设置weight属性：

```
setLayoutParams(new  LayoutParams(LayoutParams.FILL_PARENT,  Layout
```

3.为LinearLayout设置分割线

很多界面开发中都会设置一些下划线,或者分割线,从而使得界面更加整洁美观,比如下面的酷狗 音乐的注册页面:

对于这种线,我们通常的做法有两种 ①直接在布局中添加一个**view**,这个view的作用仅仅是显示出一条线,代码也很简单:

```
<View   android:layout_width="match_parent"   android:layout_height='
```

这个是水平方向上的黑线,当然你也可以改成其他颜色,或者使用图片

②第二种则是使用**LinearLayout**的一个**divider**属性,直接为LinearLayout设置分割线 这里就需要你自己准备一张线的图片了 1)android:divider设置作为分割线的图片 2)android:showDividers设置分割线的位置,none(无),begining(开始),end(结束),middle(每两个组件间) 3)dividerPadding设置分割线的Padding

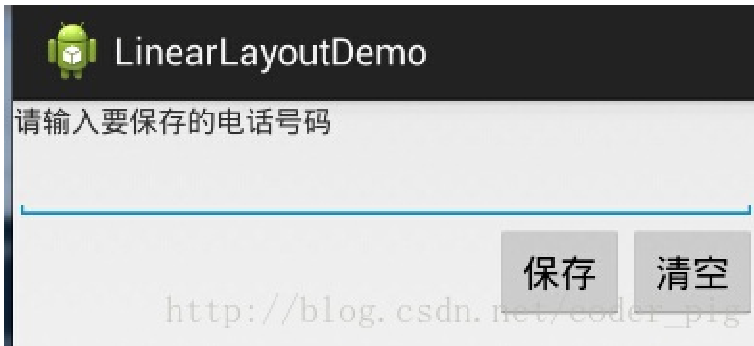
使用示例：



实现代码：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

4.LinearLayout的简单例子:



实现代码如下：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

5.注意事项:

使用Layout_gravity的一个很重要的问题!!! 问题内容: 在一个LinearLayout的水平方向中布置两个TextView,想让一个左,一个右,怎么搞? 或许你会脱口而出:"gravity设置一个left,一个right就可以啦!" 真的这么简单?你试过吗?写个简单的Layout你就会发现,事与愿违了: 代码如下：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

运行结果图:



看到这里你会说:哎呀,真的不行耶,要不在外层LinearLayout加个gravity=left的属性,然后设置第二个 TextView的layout_gravity为right,恩,好我们试一下:

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

结果还是一样：



好吧,没辙了,怎么办好?

> 当 **android:orientation="vertical"** 时， 只有水平方向的设置才起作用，垂直方向的设置不起作用。 即：**left**，**right**，**center_horizontal** 是生效的。 当 **android:orientation="horizontal"** 时， 只有垂直方向的设置才起作用，水平方向的设置不起作用。 即：**top**，**bottom**，**center_vertical** 是生效的。

然而，这方法好像并没有什么卵用。比如： 如果只能竖直方向设置左右对齐的话,就会出现下面的效果:



这显然不是我们要的结果把! 综上,要么按照上述给出的规则来布局,不过对于这种情况还是使用相对布局RelativeLayout把！ 网上没给出具体的原因,都是说这样改有人说这个和orientation的优先级有关，暂且先mark下来吧,后续如果知道原因的话再解释!前面屏幕适配也说过了,布局还是建议使用 **RelativeLayout!**

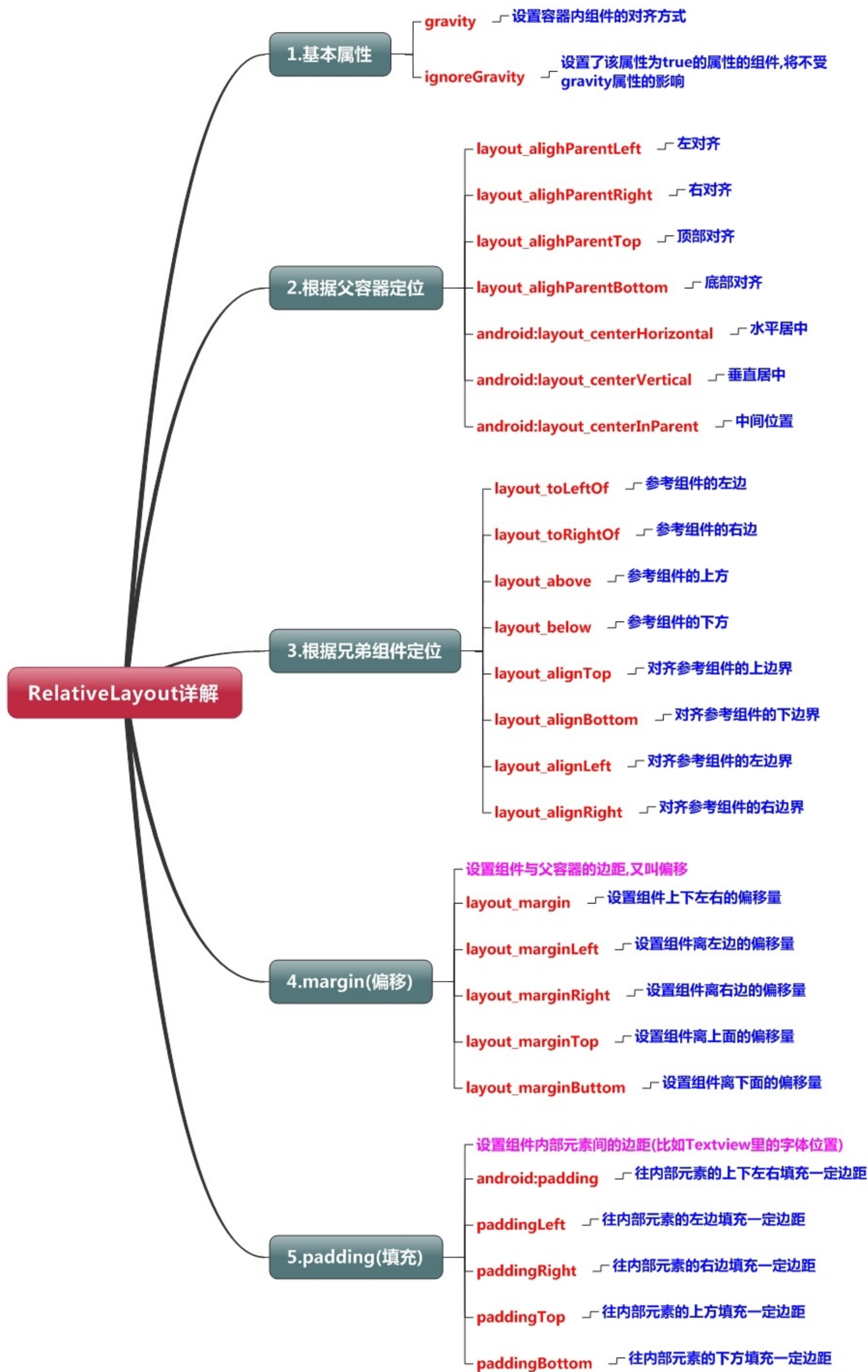
PS：本文是之前笔者写的布局系列文章中的一篇，自觉的写得很赞，所以这里直接引用的原文内容， 原文链接如下：New UI-布局之LinearLayout(线性布局)详解

2.2.2 RelativeLayout(相对布局)

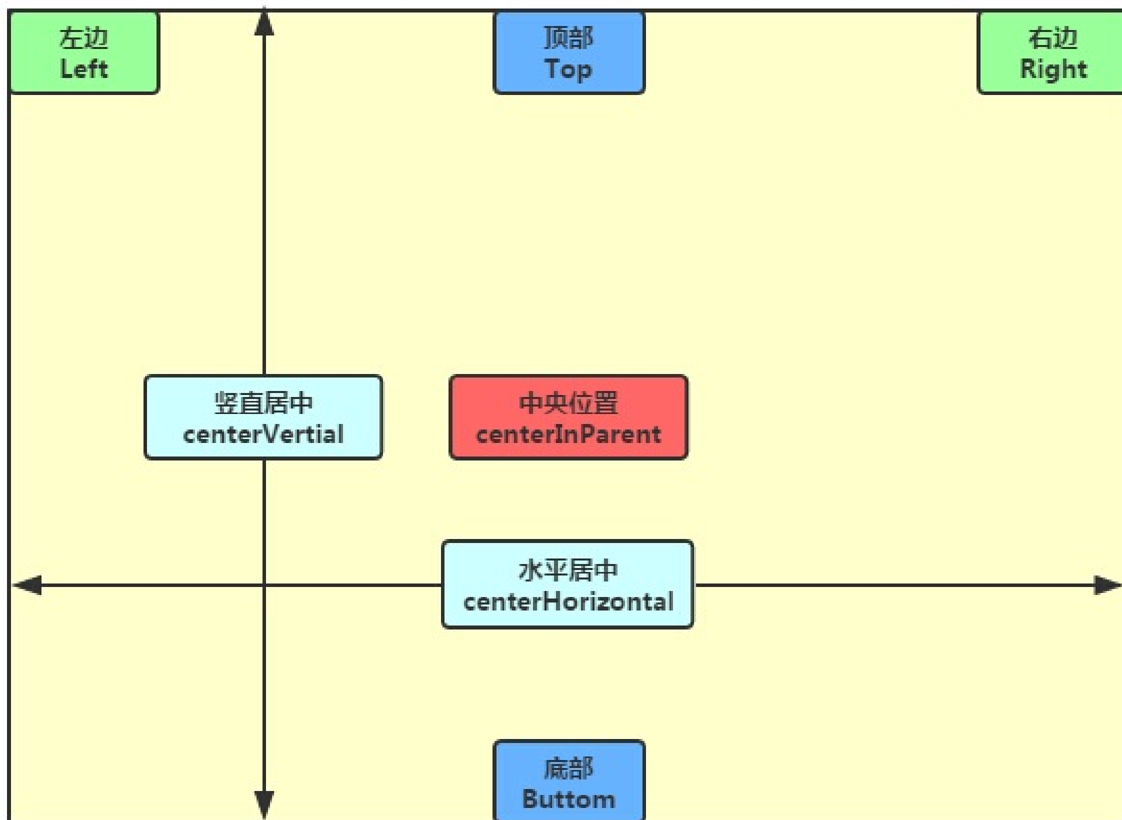
本节引言

在上一节中我们对LinearLayout进行了详细的解析，LinearLayout也是我们 用的比较多的一个布局,我们更多的时候更钟情于他的weight(权重)属性，等比例划分，对屏幕适配还是 帮助蛮大的;但是使用LinearLayout的时候也有一个问题，就是当界面比较复杂的时候，需要嵌套多层的 LinearLayout,这样就会降低UI Render的效率(渲染速度),而且如果是listview或者GridView上的 item,效率会更低,另外太多层LinearLayout嵌套会占用更多的系统资源,还有可能引发stackoverflow; 但是如果我们使用RelativeLayout的话,可能仅仅需要一层就可以完成了,以父容器或者兄弟组件参考+margin +padding就可以设置组件的显示位置,是比较方便的!当然,也不是绝对的,具体问题具体分析吧! 总结就是:尽量使用**RelativeLayout + LinearLayout**的**weight**属性搭配使用吧！

1.核心属性图



2.父容器定位属性示意图



根据父容器定位示意图 blog.csdn.net/coder_pig

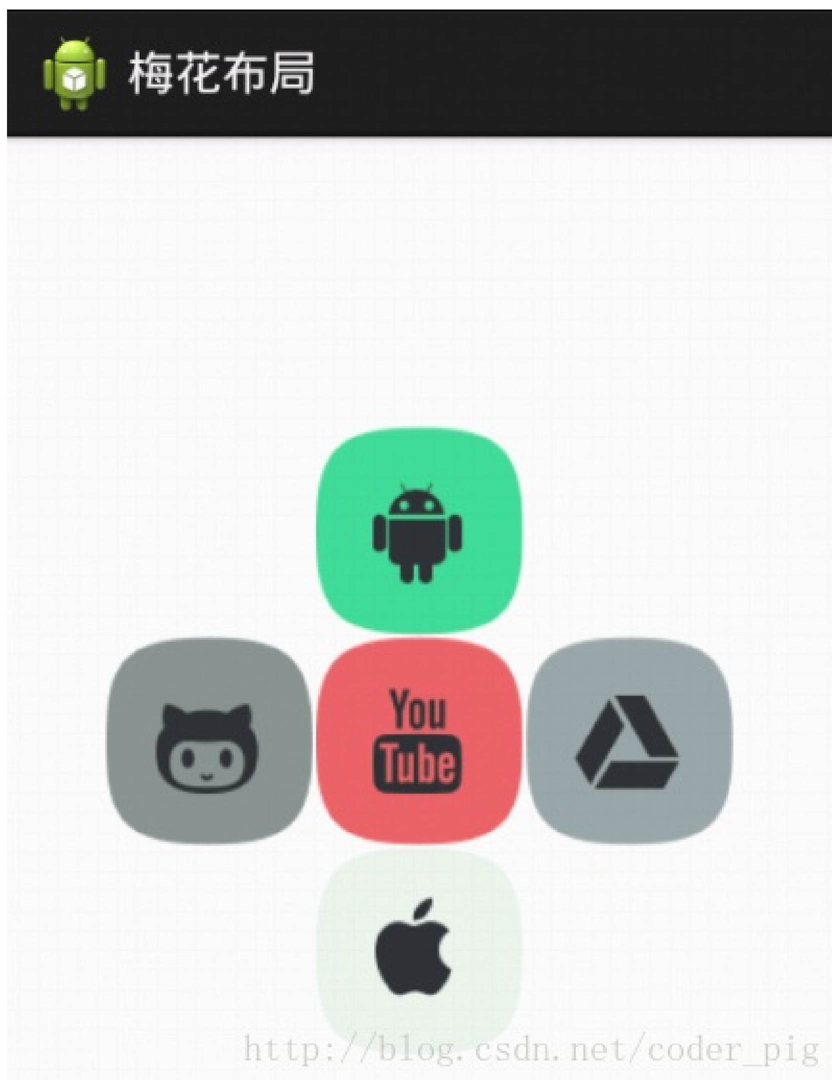
3.根据兄弟组件定位

恩,先说下什么是兄弟组件吧,所谓的兄弟组件就是处于同一层次容器的组件,如图



图中的组件1,2就是兄弟组件了,而组件3与组件1或组件2并不是兄弟组件,所以组件3不能通过 组件1或2来进行定位,比如layout_toleftof = "组件1"这样是会报错的！切记！ 关于这个兄弟组件定位的最经典例子就是"梅花布局"了,下面代码实现下:

运行效果图：



实现代码：

```
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/
```

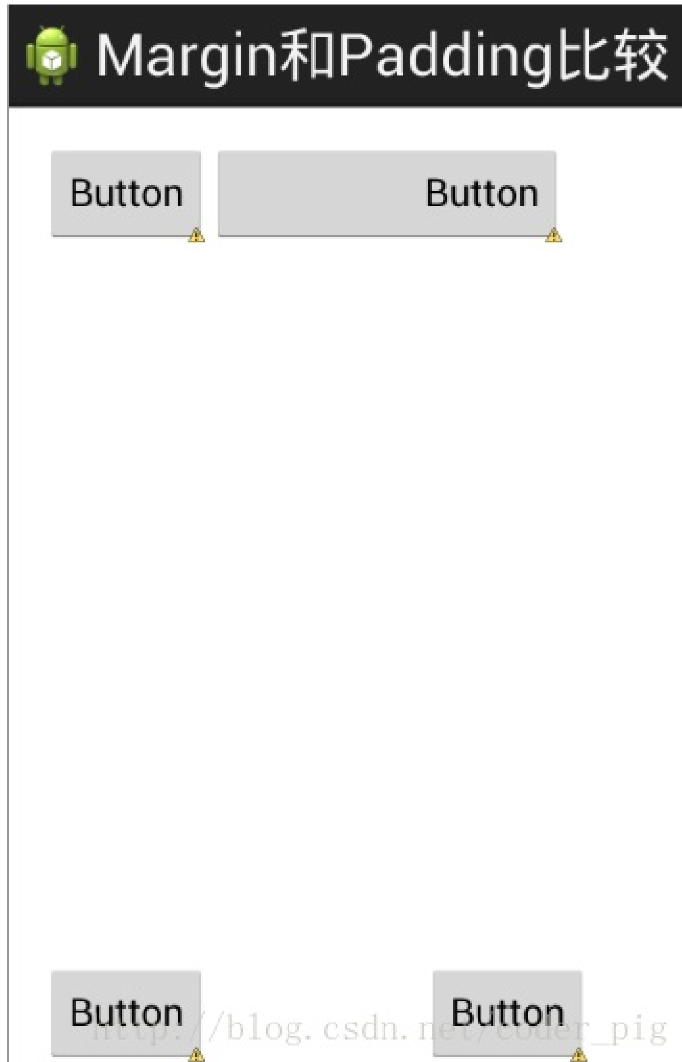
4.margin与padding的区别

初学者对于这两个属性可能会有一点混淆，这里区分下：首先margin代表的是偏移,比如marginleft = "5dp"表示组件离容器左边缘偏移5dp; 而padding代表的则是填充,而填充的对象针对的是组件中的元素,比如TextView中的文字 比如为TextView设置paddingleft = "5dp",则是在组件里的元素的左边填充5dp的空间！ margin针对的是容器中的组件，而padding针对的是组件中的元素，要区分开来！ 下面通过简单的代码演示两者的区别:

比较示例代码如下：

```
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/
```

运行效果图比较：



5.很常用的一点:margin可以设置为负数

相信很多朋友都不知道一点吧，平时我们设置margin的时候都习惯了是正数的, 其实是可以用负数的,下面写个简单的程序演示下吧,模拟进入软件后,弹出广告页面的, 右上角的cancle按钮的margin则是使用负数的！

效果图如下:



贴出的广告Activity的布局代码吧,当然,如果你对这个有兴趣的话可以下下demo, 因为仅仅是实现效果,所以代码会有些粗糙！

```
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/
```

本节小结：

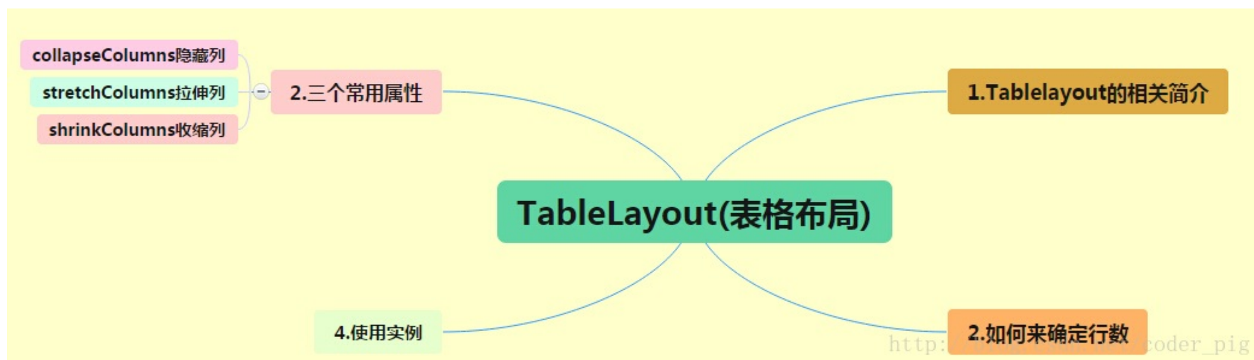
关于RelativeLayout的详解就到这里，有什么纰漏，错误，好的建议，欢迎提出~ 最后提供下上面的demo代码供大家下载：RelativeLayoutDemo

2.2.3 TableLayout(表格布局)

本节引言：

前面我们已经学习了平时实际开发中用得较多的线性布局(LinearLayout)与相对布局(RelativeLayout), 其实学完这两个基本就够用了,笔者在实际开发中用得比较多的也是这两个,当然作为一个好学的程序猿, 都是喜欢刨根问题的,所以虽说用得不多,但是还是有必要学习一下基本的用法的,说不定哪一天能用得上呢! 你说是吧,学多点东西没什么的,又不吃亏！好了,扯淡就扯到这里,开始这一节的学习吧,这一节我们会学习 Android中的第三个布局:TableLayout(表格布局)！

1.本节学习路线图



> 路线图分析： 从上面的路线图,可以看出TableLayout的用法还是很简单的,无非就是确定表格的行数,以及使用 那三个属性来设置每一行中的第某列的元素隐藏,拉伸,或者收缩即可！

2.TableLayout的介绍

> 相信学过HTML的朋友都知道,我们可以通过< table >< tr >< td >就可以生成一个HTML的表格, 而Android中也允许我们使用表格的方式来排列组件,就是行与列的方式,就说我们这节的TableLayout! 但却不像我们后面会讲到的Android 4.0后引入的**GridLayout(**网格**)**布局一样,直接就可以设置多少行与多少列！

3.如何确定行数与列数

> - ①如果我们直接往TableLayout中添加组件的话,那么这个组件将占满一行！！！
> - ②如果我们想一行上有多个组件的话,就要添加一个TableRow的容器,把组件都丢到里面！
> - ③tablerow中的组件个数就决定了该行有多少列,而列的宽度由该列中最宽的单元格决定
> - ④tablerow的layout_width属性,默认是fill_parent的,我们自己设置成其他的值也不会生效！！！ 但是layout_height默认是wrapten——content的,我们却可以自己设置大小！
> - ⑤整个表格布局的宽度取决于父容器的宽度(占满父容器本身)
> - ⑥有多少行就要自己数啦,一个tablerow一行,一个单独的组件也一行！多少列则是看tableRow中 的组件个数,组件最多的就是TableLayout的列数

4.三个常用属性

> **android:collapseColumns:**设置需要被隐藏的列的序号
> **android:shrinkColumns:**设置允许被收缩的列的列序号
> **android:stretchColumns:**设置运行被拉伸的列的列序号
>
> 以上这三个属性的列号都是从**0**开始算的,比如shrinkColunmns = "2",对应的是第三列！ 可以设置多个,用逗号隔开比如"0,2",如果是所有列都生效,则用**"*"**号即可 除了这三个常用属性,还有两个属性,分别就是跳格子以及合并单元格,这和HTML中的Table类似:
>
> **android:layout_column**="2":表示的就是跳过第二个,直接显示到第三个格子处,从1开始算的! **android:layout_span**="4":表示合并4个单元格,也就说这个组件占4个单元格

属性使用示例：

①collapseColumns(隐藏列)

流程:在TableRow中定义5个按钮后,接着在最外层的TableLayout中添加以下属性: android:collapseColumns = "0,2"，就是隐藏第一与第三列,代码如下:

```
<TableLayout
    android:id="@+id/TableLayout2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:collapseColumns="0,2" >

    <TableRow>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="one" />

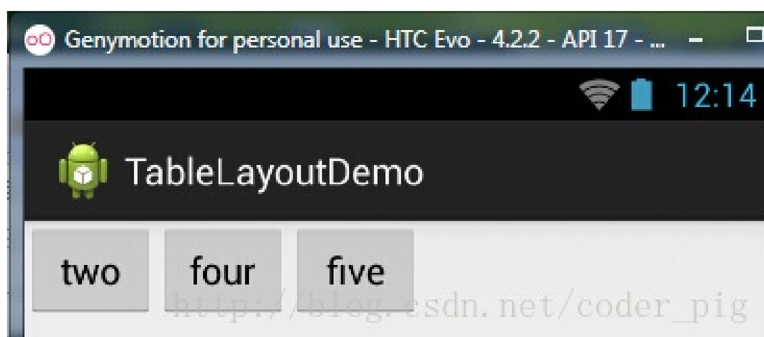
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="two" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="three" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="four" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="five" />
    </TableRow>
</TableLayout>
```

运行效果图：



②stretchColumns(拉伸列)

流程:在TableLayout中设置了四个按钮,接着在最外层的TableLayout中添加以下属性: android:stretchColumns = "1"

设置第二列为可拉伸列,让该列填满这一行所有的剩余空间,代码如下:

```
<TableLayout
    android:id="@+id/TableLayout2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1" >

    <TableRow>

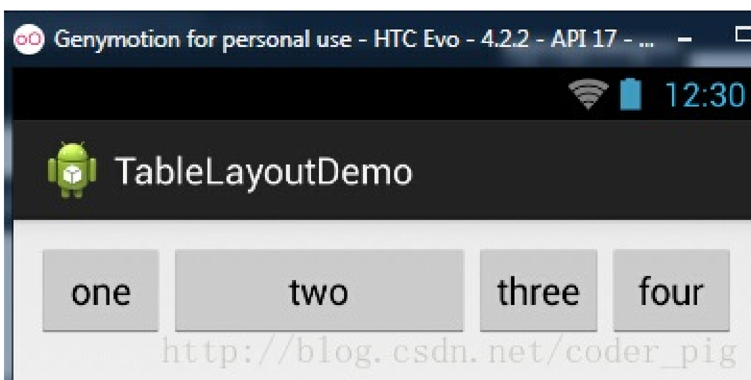
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="one" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="two" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="three" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="four" />
    </TableRow>
</TableLayout>
```

运行效果图：



③shrinkColumns(收缩列)

步骤:这里为了演示出效果,设置了5个按钮和一个文本框,在最外层的TableLayout中添加以下属性: android:shrinkColumns = "1"

设置第二个列为可收缩列,代码如下:

```
<TableLayout
    android:id="@+id/TableLayout2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:shrinkColumns="1" >

    <TableRow>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="one" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="two" />

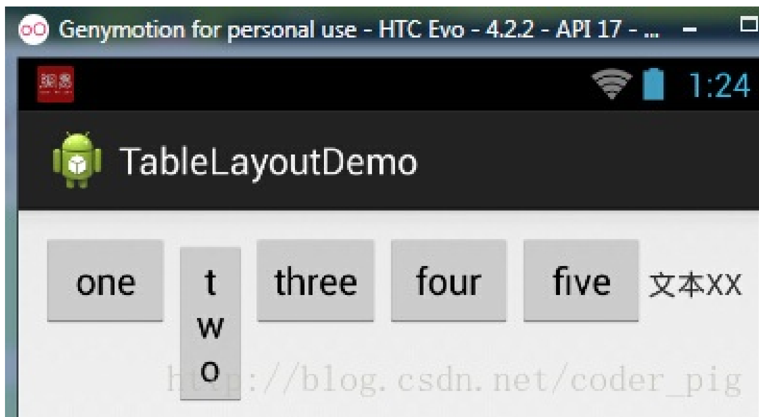
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="three" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="four" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="five" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="文本XX" />
    </TableRow>
</TableLayout>
```

运行截图:



从图中我们可以看到two这个按钮被挤压成条条状,这个就是收缩,为了保证表格能适应 父容器的宽度！至于另外两个属性就不讲解了,用法和HTML相同！有兴趣的可以研究下！

5.使用实例

使用TableLayout来完成简单的登录界面,运行效果图如下:



流程解析:

> ①调用gravity属性,设置为center_vertical,让布局里面的组件在竖直方向上居中
>
> ②将TableLayout中的第一和第四列设置为可拉伸

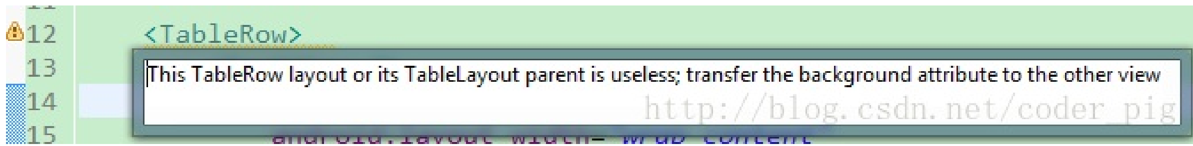
> ③在每个TableRow中添加两个TextView,用于拉伸填满该行,这样可以让表格水平居中
>
> android:stretchColumns="0,3" 设置为0.3，是为了让两边都充满,那么中间部分就可以居中了
>
> 详细代码如下：
>
> ```
> <TableLayout xmlns:android="http://schemas.android.com/apk/r
>     xmlns:tools="http://schemas.android.com/tools"
>     android:id="@+id/TableLayout1"
>     android:layout_width="match_parent"
>     android:layout_height="match_parent"
>     tools:context=".MainActivity"
>     android:stretchColumns="0,3"
>     android:gravity="center_vertical"
>     android:background="#66FF66"
>     >
>
>     <TableRow>
>         <TextView />
>         <TextView
>             android:layout_width="wrap_content"
>             android:layout_height="wrap_content"
>             android:text="用户名:"/>
>         <EditText
>             android:layout_width="wrap_content"
>             android:layout_height="wrap_content"
>             android:minWidth="150dp"/>
>         <TextView />
>     </TableRow>
>
>     <TableRow>
>         <TextView />
>         <TextView
>             android:layout_width="wrap_content"
>             android:layout_height="wrap_content"
>             android:text="密  码:"
>         />
>         <EditText
>             android:layout_width="wrap_content"
>             android:layout_height="wrap_content"
>             android:minWidth="150dp"
>         />
>         <TextView />
>     </TableRow>
>
>     <TableRow>
>         <TextView />
>         <Button
>             android:layout_width="wrap_content"
>             android:layout_height="wrap_content"
> ```

```
                android:text="登陆"/&gt;
            &lt;Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="退出"/&gt;
            &lt;TextView /&gt;
        &lt;/TableRow&gt;

    &lt;/TableLayout&gt;
```

6.发现的问题

相信大家在使用这个这TableLayout的TableRow的时候会遇到这个警告:



当然,程序还是可以运行的,不过或许你是强迫症患者,看到黄色感叹号你就不爽的话！ 而解决这个警告的方法也是很奇葩的:只要你的TableLayout里面有2个或以上的TableRow就可以了！

本节小结：

好的，关于Android的第三个布局:TableLayout就到这里~无非就是五个属性的使用而已，实际开发 表格布局我们用的不多，知道简单的用法就可以了！

2.2.4 FrameLayout(帧布局)

本节引言

FrameLayout(帧布局)可以说是六大布局中最为简单的一个布局,这个布局直接在屏幕上开辟出一块空白的区域,当我们往里面添加控件的时候,会默认把他们放到这块区域的左上角,而这种布局方式却没有任何的定位方式,所以它应用的场景并不多;帧布局的大小由控件中最大的子控件决定,如果控件的大小一样大的话,那么同一时刻就只能看到最上面的那个组件!后续添加的控件会覆盖前一个!虽然默认会将控件放置在左上角,但是我们也可以通过layout_gravity属性,指定到其他的位置!本节除了给大家演示一个最简单的例子外,还给大家带了两个好玩的例子,有兴趣的可以看看!

1.常用属性

FrameLayout的属性很少就两个,但是在说之前我们先介绍一个东西:

前景图像:永远处于帧布局最上面,直接面对用户的图像,就是不会被覆盖的图片。

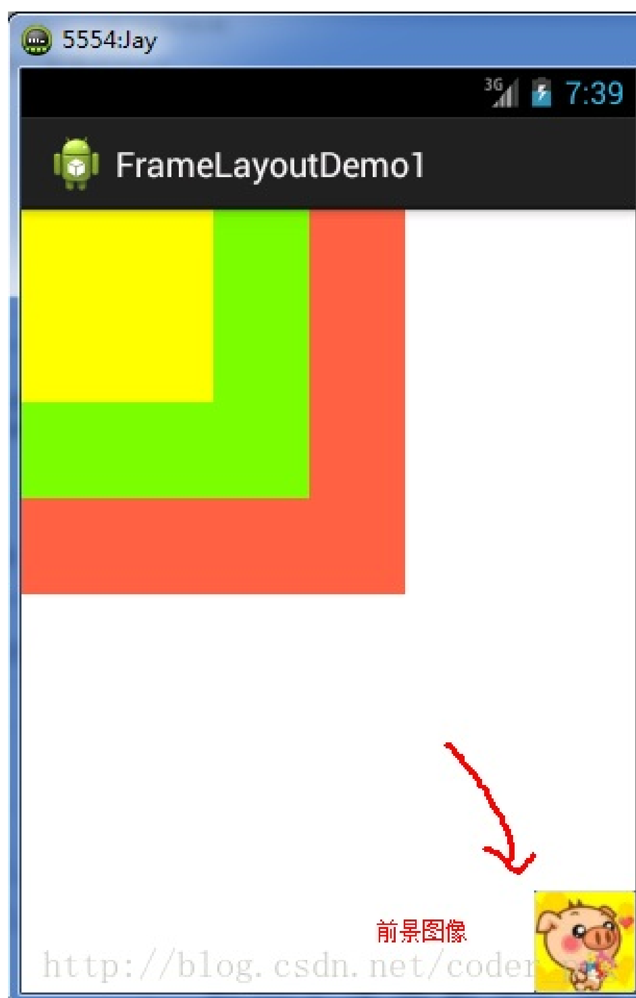
两个属性:

- **android:foreground:***设置改帧布局容器的前景图像
- **android:foregroundGravity:**设置前景图像显示的位置

2.实例演示

1)最简单的例子

运行效果图：


5554:Jay
7:39
FrameLayoutDemo1
前景图像
http://blog.csdn.net/coder

实现代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:foreground="@drawable/logo"
    android:foregroundGravity="right|bottom">

    <TextView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="#FF6143" />
    <TextView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:background="#7BFE00" />
     <TextView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="#FFFF00" />

</FrameLayout>
```

代码解析**:** 很简单,三个TextView设置不同大小与背景色,依次覆盖,接着右下角的是前景图像,通过 android:foreground="@drawable/logo"设置前景图像的图片, android:foregroundGravity="right|bottom"设置前景图像的位置在右下角

2)随手指移动的萌妹子

效果图如下：



实现流程解析：

- step 1:先将main.xml布局设置为空白的FrameLayout,为其设置一个图片背景
- step 2:新建一个继承View类的MeziView自定义组件类,在构造方法中初始化view的初始坐标
- step 3:重写onDraw()方法,实例化一个空的画笔类Paint
- step 4:调用BitmapFactory.decodeResource()生成位图对象
- step 5:调用canvas.drawBitmap()绘制妹子的位图对象
- step 6:判断图片上是否回收,否则强制回收图片
- step 7:在主Java代码中获取帧布局对象,并且实例化一个MeziView类
- step 8:会实例化的mezi对象添加一个触摸事件的监听器,重写onTouch方法,改变mezi的X,Y坐标，调用invalidate()重绘方法
- step 9: 将mezi对象添加到帧布局中

布局代码：**main_activity.xml**

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mylayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@drawable/back" >
</FrameLayout>
```

自定义的**MeziView.java**

```
package com.jay.example.framelayoutdemo2;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.view.View;

public class MeziView extends View {
    //定义相关变量,依次是妹子显示位置的X,Y坐标
    public float bitmapX;
    public float bitmapY;
    public MeziView(Context context) {
        super(context);
        //设置妹子的起始坐标
        bitmapX = 0;
        bitmapY = 200;
    }

    //重写View类的onDraw()方法
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        //创建,并且实例化Paint的对象
        Paint paint = new Paint();
        //根据图片生成位图对象
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResou
        //绘制萌妹子
        canvas.drawBitmap(bitmap, bitmapX, bitmapY,paint);
        //判断图片是否回收,木有回收的话强制收回图片
        if(bitmap.isRecycled())
        {
            bitmap.recycle();
        }
    }
}
```

MainActivity.java:

```
package com.jay.example.framelayoutdemo2;

import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.FrameLayout;
import android.app.Activity;

public class MainActivity extends Activity {

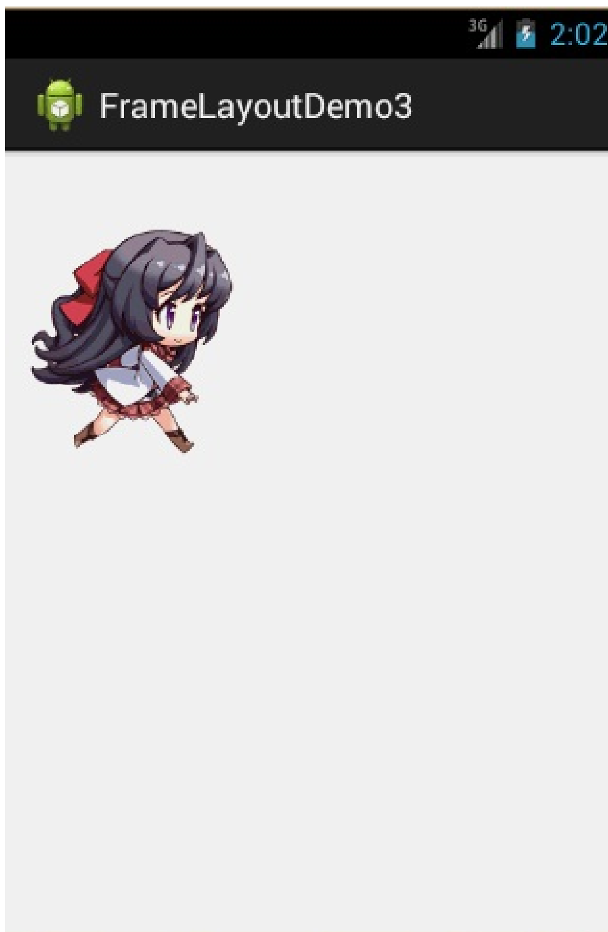
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FrameLayout frame = (FrameLayout) findViewById(R.id.mylayou
        final MeziView mezi = new MeziView(MainActivity.this);
        //为我们的萌妹子添加触摸事件监听器
        mezi.setOnTouchListener(new OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent event) {
                //设置妹子显示的位置
                mezi.bitmapX = event.getX() - 150;
                mezi.bitmapY = event.getY() - 150;
                //调用重绘方法
                mezi.invalidate();
                return true;
            }
        });
        frame.addView(mezi);
    }
}
```

代码解释: 见步骤,很简单,就是自定义一个View类,重写重绘方法,接着在Activity中为他添加一个触摸时间在触摸时间中重写onTouch方法获取点击焦点,另外还需要-150,不然那个坐标是自定义View的左上角,接着调用invalidate( )重绘方法,最后添加到帧布局中而已!

代码下载:FrameLayoutDemo2.zip

3)跑动的萌妹子

效果图如下：



实现流程:

- step 1:定义一个空的FrameLayout布局,将前景图像的位置设置为中央位置
- step 2:在Activity中获取到该FrameLayout布局,新建一个Handler对象,重写handlerMessage()方法,调用图像- 更新的方法
- step 3:自定义一个move()方法,通过switch动态设置前景图片显示的位图
- step 4:在onCreate()方法中新建一个计时器对象Timer,重写run方法,每隔170毫秒向handler发送空信息

实现代码如下:

布局文件**:main_activity.xml:**

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/myframe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:foregroundGravity="center">
</FrameLayout>
```

MainActivity.java:

```
package com.jay.example.framelayoutdemo3;

import java.util.Timer;
import java.util.TimerTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.FrameLayout;
import android.app.Activity;
import android.graphics.drawable.Drawable;

public class MainActivity extends Activity {
    //初始化变量,帧布局
    FrameLayout frame = null;
    //自定义一个用于定时更新UI界面的handler类对象
        Handler handler = new Handler()
        {
            int i = 0;
            @Override
            public void handleMessage(Message msg) {
            //判断信息是否为本应用发出的
                if(msg.what == 0x123)
                {
                    i++;
                    move(i % 8 );
                }
                super.handleMessage(msg);
             }
        };

    //定义走路时切换图片的方法
    void move(int i)
    {
        Drawable a = getResources().getDrawable(R.drawable.s_1);
        Drawable b = getResources().getDrawable(R.drawable.s_2);
        Drawable c = getResources().getDrawable(R.drawable.s_3);
        Drawable d = getResources().getDrawable(R.drawable.s_4);
        Drawable e = getResources().getDrawable(R.drawable.s_5);
        Drawable f = getResources().getDrawable(R.drawable.s_6);
        Drawable g = getResources().getDrawable(R.drawable.s_7);
        Drawable h = getResources().getDrawable(R.drawable.s_8);
        //通过setForeground来设置前景图像
        switch(i)
        {
            case 0:
                frame.setForeground(a);
                break;
            case 1:
                frame.setForeground(b);
                break;
```

```
                case 2:
                    frame.setForeground(c);
                    break;
                case 3:
                    frame.setForeground(d);
                    break;
                case 4:
                    frame.setForeground(e);
                    break;
                case 5:
                    frame.setForeground(f);
                    break;
                case 6:
                    frame.setForeground(g);
                    break;
                case 7:
                    frame.setForeground(h);
                    break;
            }
        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            frame = (FrameLayout) findViewById(R.id.myframe);
            //定义一个定时器对象,定时发送信息给handler
            new Timer().schedule(new TimerTask() {

                @Override
                public void run() {
                    //发送一条空信息来通知系统改变前景图片
                    handler.sendEmptyMessage(0x123);
                }
            }, 0,170);
        }
    }
```

代码解析: 代码也很简单,就是定义一个handler对象来刷新帧布局的前景图像,定义一个Timer定时器每隔170毫秒发送定时信息,i++;move(i%8);这里是因为我们使用8个图片作为动画素材！

代码下载:FrameLayoutDemo3.zip

本节小结

本节介绍了下FrameLayout(帧布局)，主要掌握foreground和foregroundGravity属性的使用即可！帧布局比前面的表格布局用得稍微多一点！有兴趣可以像笔者这样写点小例子试试！

2.2.5 GridLayout(网格布局)

本节引言

今天要介绍的布局是Android 4.0以后引入的一个新的布局,和前面所学的TableLayout(表格布局) 有点类似,不过他有很多前者没有的东西,也更加好用,

> - 可以自己设置布局中组件的排列方式
> - 可以自定义网格布局有多少行,多少列
> - 可以直接设置组件位于某行某列
> - 可以设置组件横跨几行或者几列

另外,除了上述内容外,本节还会给大家使用gridLayout时会遇到的问题,以及如何解决低版本 sdk如何使用GridLayout的方法！接下来就开始本节的课程吧!

1.相关属性总结图



GridLayout相关属性图

2.使用实例:计算器布局的实现:

运行效果图：



实现代码:

```
<pre>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/andro
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/GridLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:orientation="horizontal"
    android:rowCount="6" >

    <TextView
        android:layout_columnSpan="4"
        android:layout_gravity="fill"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:background="#FFCCCC"
        android:text="0"
        android:textSize="50sp" />

    <Button
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:text="回退" />

    <Button
        android:layout_columnSpan="2"
```

```
        android:layout_gravity="fill"
        android:text="清空" />

    <Button android:text="+" />

    <Button android:text="1" />

    <Button android:text="2" />

    <Button android:text="3" />

    <Button android:text="-" />

    <Button android:text="4" />

    <Button android:text="5" />

    <Button android:text="6" />

    <Button android:text="*" />

    <Button android:text="7" />

    <Button android:text="8" />

    <Button android:text="9" />

    <Button android:text="/" />

    <Button
        android:layout_width="wrap_content"
        android:text="." />

    <Button android:text="0" />

    <Button android:text="=" />

</GridLayout>
```

代码解析**:** 代码很简单,只是回退与清楚按钮横跨两列,而其他的都是直接添加的,默认每个组件都是 占一行一列,另外还有一点要注意的: 我们通过:**android:layout_rowSpan**与**android:layout_columnSpan**设置了组件横跨 多行或者多列的话,如果你要让组件填满横越过的行或列的话,需要添加下面这个属性: **android:layout_gravity = "fill"**！！！就像这个计算机显示数字的部分!

3.用法归纳:

①GridLayout使用虚细线将布局划分为行,列和单元格,同时也支持在行,列上进行交错排列 ②使用流程:

> - step 1:先定义组件的对其方式 android:orientation 水平或者竖直,设置多少行与多少列
> - step 2:设置组件所在的行或者列,记得是从0开始算的,不设置默认每个组件占一行一列
> - step 3:设置组件横跨几行或者几列;设置完毕后,需要在设置一个填充:android:layout_gravity = "fill"

4.使用GridLayout要注意的地方:

因为GirdLayout是4.0后才推出的,所以minSDK版本要改为14或者以上的版本, 不然写布局代码的时候,这玩意就会莫名其妙地出错,说找不到这个GridLayout, 当然,如果你要低版本兼容的话,就要看下面的内容了！

5.低版本sdk如何使用GridLayout:

解决方法很简单:只需要导入v7包的gridlayout包即可! v7包一般在sdk下的:sdk\extras\android\support\v7\gridlayout目录下 如果你没有的话,也可以到这里下载: gridlayout_v7_jay.rar 但是用的时候,标签却是这样写的:

```
<android.support.v7.widget.GridLayout>`
```

本节小结

关于GridLayout的介绍就到这里~

2.2.6 AbsoluteLayout(绝对布局)

本节引言

> 前面已经介绍了,Android中的五大布局,在本节中会讲解第六个布局AbsoluteLayout(绝对布局), 之所以把这个放到最后,是因为绝对布局,我们基本上都是不会使用的,当然你也可以直接跳过这一 篇博文,不过作为一个喜欢增长姿势的程序员,我们还是可以了解这个AbsoluteLayout布局的, 相信大部分学过Java的都知道,我们在Java swing(不是spring哦)都用过这个绝对布局,但是Android 中我们用这个少的原因,就是因为我们开发的应用需要在很多的机型上面进行一个适配,如果你 使用了这个绝对布局的话,可能你在4寸的手机上是显示正常的,而换成5寸的手机,就可能出现偏移 和变形,所以的话,这个还是不建议使用了,当然,如果你不会写代码的话,又想玩玩android,那么写 布局的时候就可以通过ADT把需要的组件,拖拉到界面上!这个AbsoluteLayout是直接通过X,Y坐标来 控制组件在Activity中的位置的！另外这个但单位是dp!

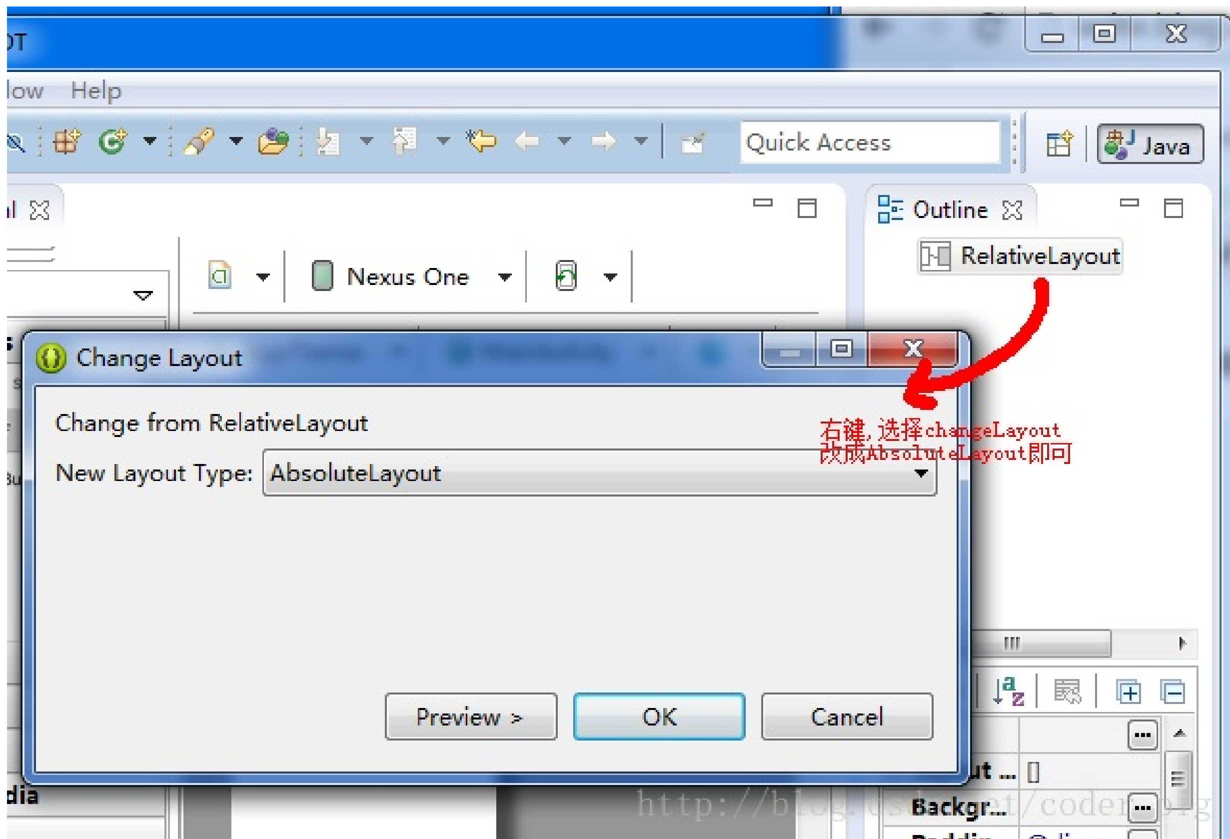
1.四大控制属性(单位都是dp):

①控制大小**: android:layout_width:**组件宽度 **android:layout_height:**组件高度 ②控制位置**: android:layout_x:**设置组件的X坐标 **android:layout_y:**设置组件的Y坐标

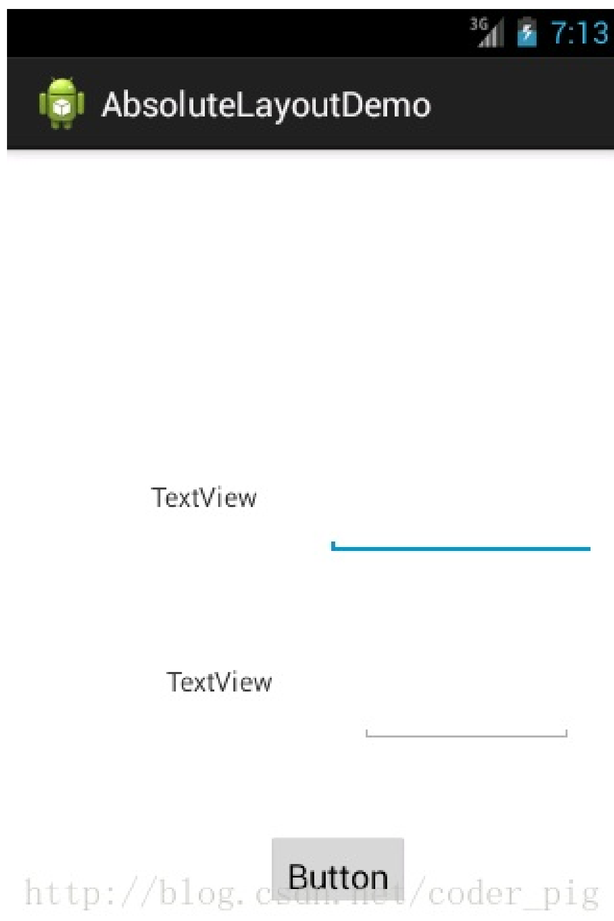
2.使用示例:

一个简单的登录界面,都是直接在ADT上拖拉出来的界面,代码就不贴了:

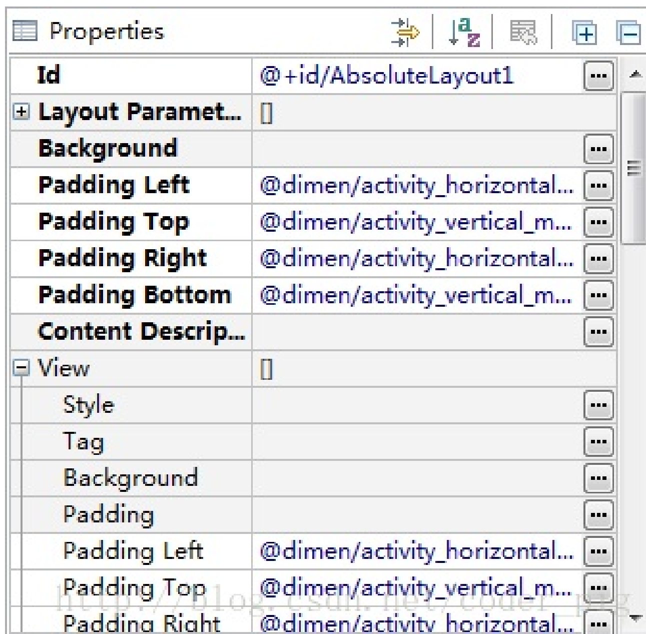
①先设置成AbsoluteLayout绝对布局:



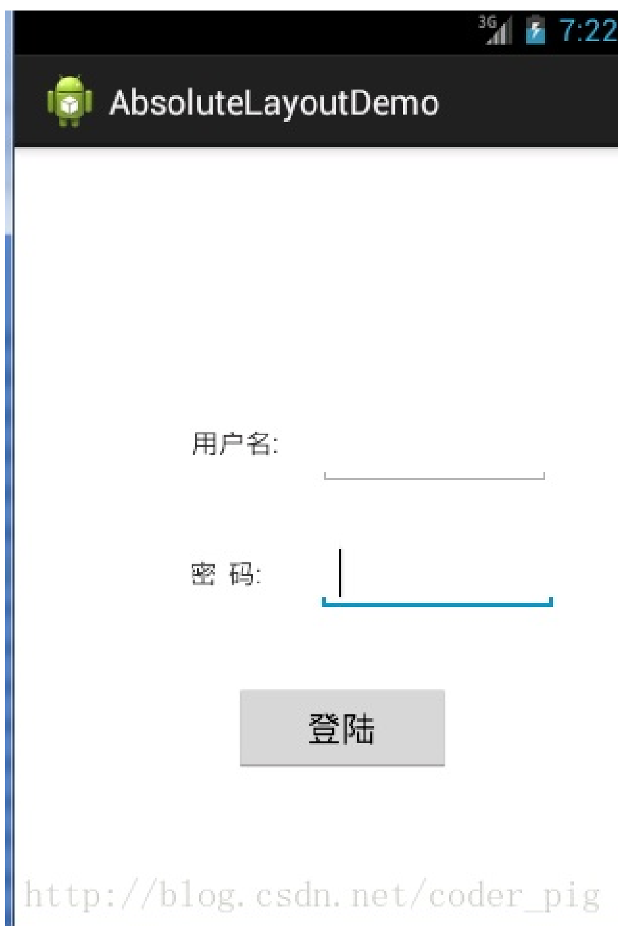
②从左边拖拉两个TextView和EditText以及一个按钮到界面上,拖来拖去,知道看上去好看点为止



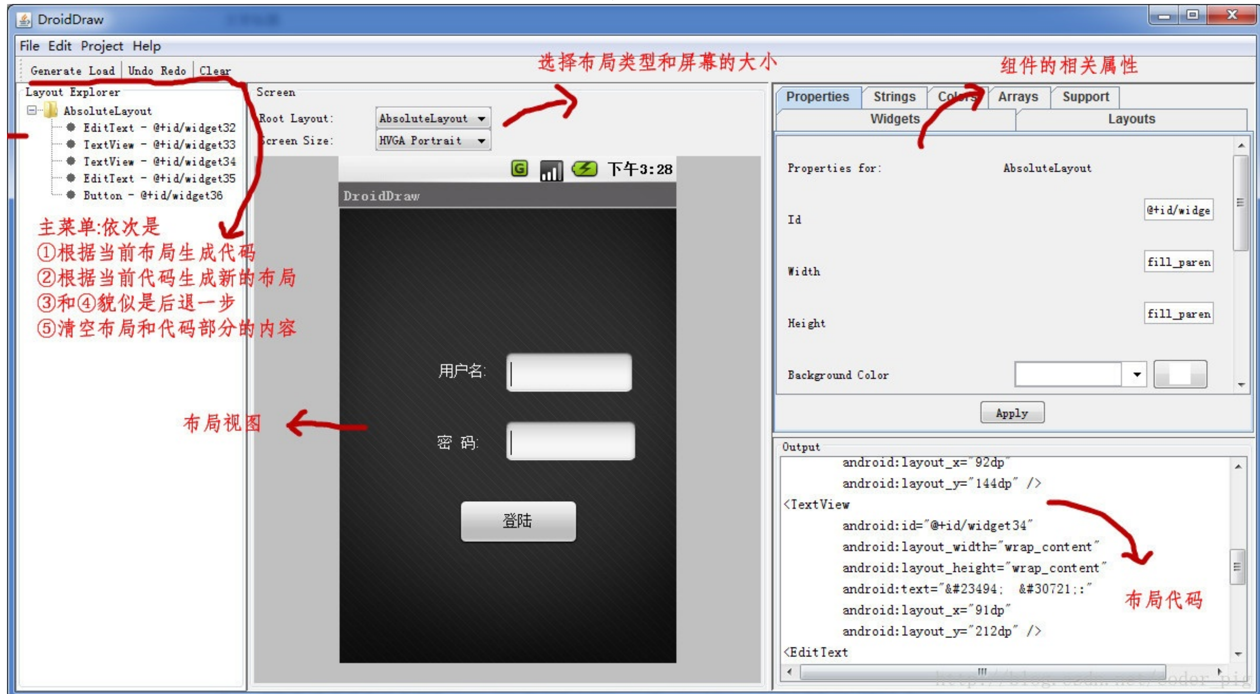
拖拉拖去,最后还是这个挫样,估计是ADT的问题,接下来你可以: ①进代码修改坐标,知道看上去好看点 ②当然,你也可以直接在右边设置坐标:



修改完后:



当然,或许以上两种方法都不喜欢,你想拖拉后就直接生成正确的代码~！ ok,没问题,推荐你使用前面界面原型那里给大家介绍的一个工具：DroidDraw



本节小结

好的，Android中的六大布局都给大家介绍完了，还是建议使用：LinearLayout的weight权重属性+ RelativeLayout来构建我们的界面~嗯，就到这里，谢谢~

2.3.1 TextView(文本框)详解

本节引言：

> 学习完Android中的六大布局，从本节开始我们来一个个讲解Android中的UI控件，本节给大家带来的UI控件是：TextView(文本框)，用于显示文本的一个控件，另外声明一点，我不是翻译API文档，不会一个个属性的去扣，只学实际开发中常用的，有用的，大家遇到感觉到陌生的属性可以查询对应的API！当然，每一节开始都会贴这一节对应API文档的链接：TextView API 好了，在开始本节内容前，先要介绍下几个单位：
>
> **dp(dip):** device independent pixels(设备独立像素). 不同设备有不同的显示效果,这个和设备硬件有关，一般我们为了支持WVGA、HVGA和QVGA 推荐使用这个，不依赖像素。 **px: pixels(**像素**)**. 不同设备显示效果相同，一般我们HVGA代表320x480像素，这个用的比较多。 **pt: point**，是一个标准的长度单位，1pt＝1/72英寸，用于印刷业，非常简单易用； **sp: scaled pixels(**放大像素**)**. 主要用于字体显示best for textsize。

1.基础属性详解：

通过下面这个简单的界面，我们来了解几个最基本的属性：



布局代码：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:gravity="center"
    android:background="#8fffad">

    <TextView
        android:id="@+id/txtOne"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:gravity="center"
        android:text="TextView(显示框)"
        android:textColor="#EA5246"
        android:textStyle="bold|italic"
        android:background="#000000"
        android:textSize="18sp" />

</RelativeLayout>
```

上面的TextView中有下述几个属性:

- **id**：为TextView设置一个组件id，根据id，我们可以在Java代码中通过findViewById()的方法获取到该对象，然后进行相关属性的设置，又或者使用RelativeLayout时，参考组件用的也是id！
- **layout_width**：组件的宽度，一般写：**wrap_content**或者**match_parent(fill_parent)**，前者是控件显示的内容多大，控件就多大，而后者会填满该控件所在的父容器；当然也可以设置成特定的大小，比如我这里为了显示效果，设置成了200dp。
- **layout_height**：组件的宽度，内容同上。
- **gravity**：设置控件中内容的对齐方向，TextView中是文字，ImageView中是图片等等。
- **text**：设置显示的文本内容，一般我们是把字符串写到string.xml文件中，然后通过@String/xxx取得对应的字符串内容的，这里为了方便我直接就写到""里，不建议这样写！！！
- **textColor**：设置字体颜色，同上，通过colors.xml资源来引用，别直接这样写！
- **textStyle**：设置字体风格，三个可选值：**normal**(无效果)，**bold**(加粗)，**italic**(斜体)
- **textSize**：字体大小，单位一般是用sp！
- **background**：控件的背景颜色，可以理解为填充整个控件的颜色，可以是图片哦！

2.实际开发例子

2.1 带阴影的TextView

涉及到的几个属性：

- **android:shadowColor:**设置阴影颜色,需要与shadowRadius一起使用哦!
- **android:shadowRadius:**设置阴影的模糊程度,设为0.1就变成字体颜色了,建议使用3.0
- **android:shadowDx:**设置阴影在水平方向的偏移,就是水平方向阴影开始的横坐标位置
- **android:shadowDy:**设置阴影在竖直方向的偏移,就是竖直方向阴影开始的纵坐标位置

效果图：



实现代码：

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:shadowColor="#F9F900"
        android:shadowDx="10.0"
        android:shadowDy="10.0"
        android:shadowRadius="3.0"
        android:text="带阴影的TextView"
        android:textColor="#4A4AFF"
        android:textSize="30sp" />
```

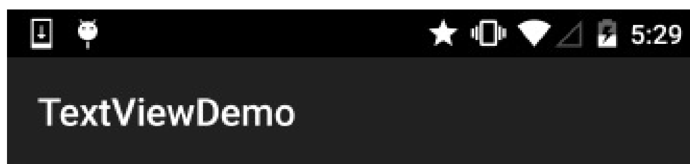
2.2 带边框的TextView：

> 如果你想为TextView设置一个边框背景，普通矩形边框或者圆角边框！下面可能帮到你！ 另外TextView是很多其他控件的父类，比如Button，也可以设置这样的边框！ 实现原理很简单，自行编写一个ShapeDrawable的资源文件！然后TextView将blackgroung 设置为这个drawable资源即可！

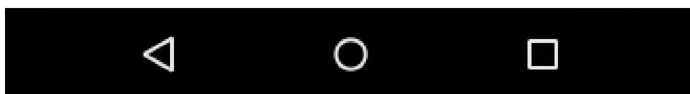
简单说下shapeDrawable资源文件的几个节点以及属性：

> - <**solid** android:color = "xxx"> 这个是设置背景颜色的
> - <**stroke** android:width = "xdp" android:color="xxx"> 这个是设置边框的粗细,以及边框颜色的
> - <**padding** androidLbottom = "xdp"...> 这个是设置边距的
> - <**corners** android:topLeftRadius="10px"...> 这个是设置圆角的
> - <**gradient**> 这个是设置渐变色的,可选属性有: **startColor**:起始颜色 **endColor**:结束颜色 **centerColor**:中间颜色 **angle**:方向角度,等于0时,从左到右,然后逆时针方向转,当angle = 90度时从下往上 **type**:设置渐变的类型

实现效果图：







代码实现：

Step 1:编写矩形边框的Drawable：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >

    <!-- 设置一个黑色边框 -->
    <stroke android:width="2px" android:color="#000000"/>
    <!-- 渐变 -->
    <gradient
        android:angle="270"
        android:endColor="#C0C0C0"
        android:startColor="#FCD209" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:left="5dp"
        android:top="5dp"
        android:right="5dp"
        android:bottom="5dp"/>

</shape>
```

Step 2:编写圆角矩形边框的Drawable：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- 设置透明背景色 -->
    <solid android:color="#87CEEB" />

    <!-- 设置一个黑色边框 -->
    <stroke
        android:width="2px"
        android:color="#000000" />
    <!-- 设置四个圆角的半径 -->
    <corners
        android:bottomLeftRadius="10px"
        android:bottomRightRadius="10px"
        android:topLeftRadius="10px"
        android:topRightRadius="10px" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:bottom="5dp"
        android:left="5dp"
        android:right="5dp"
        android:top="5dp" />

</shape>
```

Step 3:将TextView的blackground属性设置成上面这两个Drawable：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtOne"
        android:layout_width="200dp"
        android:layout_height="64dp"
        android:textSize="18sp"
        android:gravity="center"
        android:background="@drawable/txt_rectborder"
        android:text="矩形边框的TextView" />

    <TextView
        android:id="@+id/txtTwo"
        android:layout_width="200dp"
        android:layout_height="64dp"
        android:layout_marginTop="10dp"
        android:textSize="18sp"
        android:gravity="center"
        android:background="@drawable/txt_radiuborder"
        android:text="圆角边框的TextView" />

</LinearLayout>
```

2.3 带图片(drawableXxx)的TextView：

在实际开发中，我们可能会遇到这种需求：



如图，要实现这种效果，可能你的想法是：一个ImageView用于显示图片 + 一个TextView用于显示文字，然后把他们丢到一个LinearLayout中，接着依次创建四个这样的小布局，再另外放到一个大的LinearLayout中，效果是可以实现，但是会不会有点繁琐呢？而且前面我们前面也说过，布局层次越少，性能越好！使用drawableXxx就可以省掉上面的过程，直接设置四个TextView就可以完成我们的需求！

基本用法：

设置图片的核心其实就是:**drawableXxx**;可以设置四个方向的图片: **drawableTop**(上),**drawableButtom**(下),**drawableLeft**(左),**drawableRight**(右) 另外,你也可以使用**drawablePadding**来设置图片与文字间的间距！

效果图：(设置四个方向上的图片)



实现代码：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.jay.example.test.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawableTop="@drawable/show1"
        android:drawableLeft="@drawable/show1"
        android:drawableRight="@drawable/show1"
        android:drawableBottom="@drawable/show1"
        android:drawablePadding="10dp"
        android:text="张全蛋" />

</RelativeLayout>
```

一些问题： 可能你会发现，我们这样设置的drawable并不能自行设置大小，在XML是无法直接设置的； 所以我们需要在Java代码中来进行一个修改！

示例代码如下：

```
package com.jay.example.test;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {
    private TextView txtZQD;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtZQD = (TextView) findViewById(R.id.txtZQD);
        Drawable[] drawable = txtZQD.getCompoundDrawables();
        // 数组下表0~3,依次是:左上右下
        drawable[1].setBounds(100, 0, 200, 200);
        txtZQD.setCompoundDrawables(drawable[0], drawable[1], drawa
                drawable[3]);
    }
}
```

运行效果图：

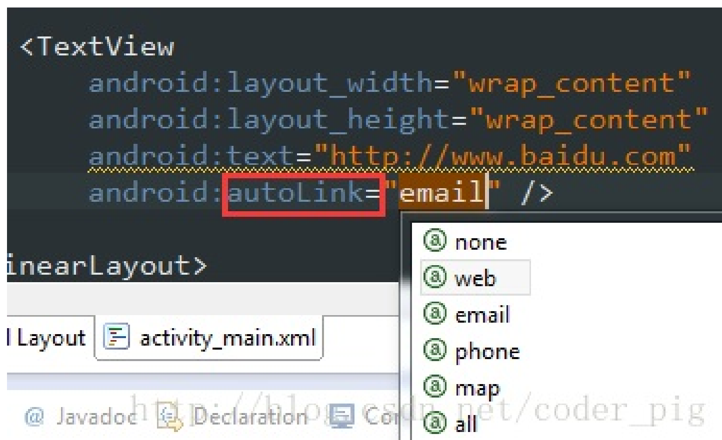


代码分析：

> - ①Drawable[] drawable = txtZQD.getCompoundDrawables( ); 获得四个不同方向上的图片资源,数组元素依次是:左上右下的图片
> - ②drawable[1].setBounds(100, 0, 200, 200); 接着获得资源后,可以调用setBounds设置左上右下坐标点,比如这里设置了代表的是: 长是:从离文字最左边开始100dp处到200dp处 宽是:从文字上方0dp处往上延伸200dp!
> - ③txtZQD.setCompoundDrawables(drawable[0], drawable[1], drawable[2], drawable[3]);为TextView重新设置drawable数组!没有图片可以用null代替哦! PS：另外，从上面看出我们也可以直接在Java代码中调用setCompoundDrawables为 TextView设置图片！

2.4 使用autoLink属性识别链接类型

当文字中出现了URL，E-Mail，电话号码，地图的时候，我们可以通过设置autoLink属性；当我们点击 文字中对应部分的文字，即可跳转至某默认APP，比如一串号码，点击后跳转至拨号界面！



看下效果图：



all就是全部都包含,自动识别协议头~ 在Java代码中可以调用setAutoLinkMask(Linkify.ALL); 这个时候可以不写协议头,autolink会自动识别，但是还要为这个TextView设置：setMovementMethod(LinkMovementMethod.getInstance()); 不然点击了是没效果的！

2.5 TextView玩转HTML

> 如题，除了显示普通文本外，TextView还预定义了一些类似于HTML的标签，通过这些标签，我们可以使 TextView显示不同的字体颜色，大小，字体，甚至是显示图片，或者链接等！我们只要使用HTML中的一些 标签，加上 android.text.HTML类的支持，即可完成上述功能！

PS:当然，并不是支持所有的标签，常用的有下述这些：

> - **<font>**：设置颜色和字体。
> - **<big>**：设置字体大号
> - **<small>**：设置字体小号
> - **<i><b>**：斜体粗体
> - **<a>**：连接网址
> - **<img>**：图片

如果直接setText的话是没作用的，我们需要调用Html.fromHtml()方法将字符串转换为CharSequence接口， 然后再进行设置，如果我们需要相应设置，需要为TextView进行设置，调用下述方法：
`Java setMovementMethod(LinkMovementMethod.getInstance())`

嗯，接着我们写代码来试试：

1）测试文本与超链接标签

```
package jay.com.example.textviewdemo;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.text.method.LinkMovementMethod;
import android.text.util.Linkify;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView t1 = (TextView)findViewById(R.id.txtOne);
        String s1 = "<font color='blue'><b>百度一下，你就知道~：</b></
        s1 += "<a href = 'http://www.baidu.com'>百度</a>";
        t1.setText(Html.fromHtml(s1));
        t1.setMovementMethod(LinkMovementMethod.getInstance());
    }
}
```

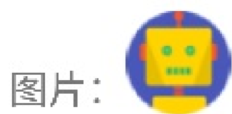
运行效果图：



恩呢，测试完毕~

2）测试src标签，插入图片：

看下运行效果图：



接下来看下实现代码，实现代码看上去有点复杂，用到了反射(对了，别忘了在drawable目录下放一个icon的图片哦！)：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView t1 = (TextView) findViewById(R.id.txtOne);
        String s1 = "图片：<img src = 'icon'/><br>";
        t1.setText(Html.fromHtml(s1, new Html.ImageGetter() {
            @Override
            public Drawable getDrawable(String source) {
                Drawable draw = null;
                try {
                    Field field = R.drawable.class.getField(source)
                    int resourceId = Integer.parseInt(field.get(nul
                    draw = getResources().getDrawable(resourceId);
                    draw.setBounds(0, 0, draw.getIntrinsicWidth(),
                } catch (Exception e) {
                    e.printStackTrace();
                }
                return draw;
            }
        }, null));
    }
}
```

嘿嘿，你也可以自己试试，比如为图片加上超链接，点击图片跳转这样~

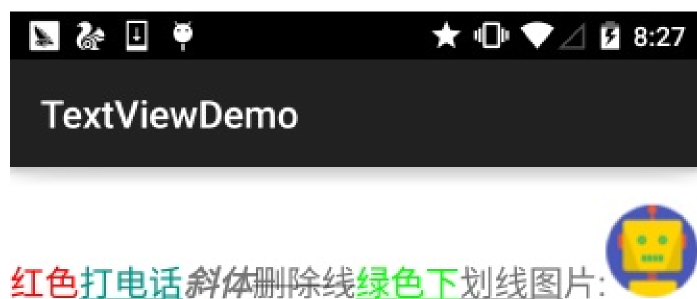
2.6 SpannableString&SpannableStringBuilder定制文本

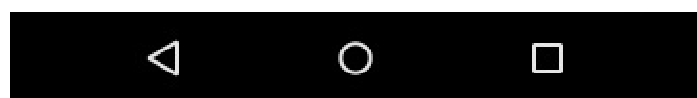
> 除了上面的HTML可以定制我们TextView的样式外，还可以使用SpannableString和SpannableStringBuilder来完成，两者区别：前者针对的是不可变文本，而后者则是针对可变文本，这里只讲解前者，对后者有兴趣可自行查阅文本！

SpannableString可供我们使用的API有下面这些：

> - **BackgroundColorSpan** 背景色
> - **ClickableSpan** 文本可点击，有点击事件
> - **ForegroundColorSpan** 文本颜色（前景色）
> - **MaskFilterSpan** 修饰效果，如模糊(BlurMaskFilter)、浮雕(EmbossMaskFilter)
> - **MetricAffectingSpan** 父类，一般不用
> - **RasterizerSpan** 光栅效果
> - **StrikethroughSpan** 删除线（中划线）
> - **SuggestionSpan** 相当于占位符
> - **UnderlineSpan** 下划线
> - **AbsoluteSizeSpan** 绝对大小（文本字体）
> - **DynamicDrawableSpan** 设置图片，基于文本基线或底部对齐。
> - **ImageSpan** 图片
> - **RelativeSizeSpan** 相对大小（文本字体）
> - **ReplacementSpan** 父类，一般不用
> - **ScaleXSpan** 基于x轴缩放
> - **StyleSpan** 字体样式：粗体、斜体等
> - **SubscriptSpan** 下标（数学公式会用到）
> - **SuperscriptSpan** 上标（数学公式会用到）
> - **TextAppearanceSpan** 文本外貌（包括字体、大小、样式和颜色）
> - **TypefaceSpan** 文本字体
> - **URLSpan** 文本超链接

好吧，还是蛮多的，这里给出个最简单的例子吧，其他的参数调用可自行百度谷歌~ **1**）最简单例子：运行效果图：





实现代码：

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView t1 = (TextView) findViewById(R.id.txtOne);
        TextView t2 = (TextView) findViewById(R.id.txtTwo);

        SpannableString span = new SpannableString("红色打电话斜体删除
        //1.设置背景色,setSpan时需要指定的flag,Spanned.SPAN_EXCLUSIVE_
        span.setSpan(new ForegroundColorSpan(Color.RED), 0, 2, Spar
        //2.用超链接标记文本
        span.setSpan(new URLSpan("tel:4155551212"), 2, 5, Spanned.S
        //3.用样式标记文本（斜体）
        span.setSpan(new StyleSpan(Typeface.BOLD_ITALIC), 5, 7, Spa
        //4.用删除线标记文本
        span.setSpan(new StrikethroughSpan(), 7, 10, Spanned.SPAN_E
        //5.用下划线标记文本
        span.setSpan(new UnderlineSpan(), 10, 16, Spanned.SPAN_EXCL
        //6.用颜色标记
        span.setSpan(new ForegroundColorSpan(Color.GREEN), 10, 13,S
        //7.//获取Drawable资源
        Drawable d = getResources().getDrawable(R.drawable.icon);
        d.setBounds(0, 0, d.getIntrinsicWidth(), d.getIntrinsicHeig
        //8.创建ImageSpan,然后用ImageSpan来替换文本
        ImageSpan imgspan = new ImageSpan(d, ImageSpan.ALIGN_BASELI
        span.setSpan(imgspan, 18, 19, Spannable.SPAN_INCLUSIVE_EXCL
        t1.setText(span);
    }
}
```

2）实现部分可点击的**TextView** 相信玩过QQ空间和微信朋友圈的朋友对下面的东东并不陌生吧，我们可以点击 对应的用户然后进入查看用户相关的信息是吧！



下面我们就来写个简单的例子来实现下效果：

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView t1 = (TextView) findViewById(R.id.txtOne);

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 20; i++) {
            sb.append("好友" + i + ", ");
        }

        String likeUsers = sb.substring(0, sb.lastIndexOf(", ")).tc
        t1.setMovementMethod(LinkMovementMethod.getInstance());
        t1.setText(addClickPart(likeUsers), TextView.BufferType.SPA
    }

    //定义一个点击每个部分文字的处理方法
    private SpannableStringBuilder addClickPart(String str) {
        //赞的图标，这里没有素材，就找个笑脸代替下~
        ImageSpan imgspan = new ImageSpan(MainActivity.this, R.draw
        SpannableString spanStr = new SpannableString("p.");
```

```
        spanStr.setSpan(imgspan, 0, 1, Spannable.SPAN_INCLUSIVE_EX(

        //创建一个SpannableStringBuilder对象，连接多个字符串
        SpannableStringBuilder ssb = new SpannableStringBuilder(spa
        ssb.append(str);
        String[] likeUsers = str.split(", ");
        if (likeUsers.length > 0) {
            for (int i = 0; i < likeUsers.length; i++) {
                final String name = likeUsers[i];
                final int start = str.indexOf(name) + spanStr.lengt
                ssb.setSpan(new ClickableSpan() {
                    @Override
                    public void onClick(View widget) {
                        Toast.makeText(MainActivity.this, name,
                                Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void updateDrawState(TextPaint ds) {
                        super.updateDrawState(ds);
                        //删除下划线，设置字体颜色为蓝色
                        ds.setColor(Color.BLUE);
                        ds.setUnderlineText(false);
                    }
                },start,start + name.length(),0);
            }
        }
    return ssb.append("等" + likeUsers.length + "个人觉得很赞");
    }
}
```

运行效果图：

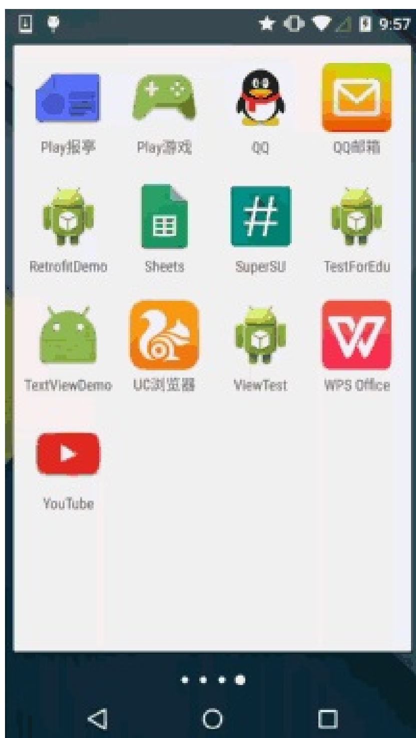


核心其实就是：ClickableSpan的设置而已~你可以自己捣鼓着写下QQ空间评论的那个自己写一个~

2.7 实现跑马灯效果的TextView

简单说下什么是跑马灯，就是类似于web一样，有一行字一直循环滚滚动这样，好吧还是看看 实现效果图，一看就懂的了~

实现效果图：



代码实现：

```
<TextView
        android:id="@+id/txtOne"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:singleLine="true"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="marquee_forever"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:text="你整天说着日了狗日了狗，但是你却没有来，呵呵呵呵呵呵呵
```

2.8 设置TextView字间距和行间距

就像我们平时编写文档的时候，我们需要排版，设置下行或者字之间的间距是吧：Android中的TextView也可以进行这样的设置：

字间距：

```
android:textScaleX：调节字间距的，默认值1.0f，值是float
Java中setScaleX(2.0f);
```

行间距： Android系统中TextView默认显示中文时会比较紧凑，为了让每行保持的行间距

> **android:lineSpacingExtra**：设置行间距，如"3dp"
> **android:lineSpacingMultiplier**：设置行间距的倍数，如"1.2"

Java代码中可以通过: **setLineSpacing**方法来设置

2.9 自动换行

先说下，如果一行显示不完，可以用：

```
android:singleLine = "true"
```

除此之外，可以也设置多行显示不完，添加个maxLines的属性即可！

3.本节小结：

本节对Android中的TextView控件进行了详细的解析，提供了开发中常见的一些问题的解决方法，相信 会为你的实际开发带来大大的便利， 另外，笔者能力有限，写出来的东西可能有些纰漏，欢迎指出， 不胜感激~另外，转载请注明出处：coder-pig！谢谢~

2.3.2 EditText(输入框)详解

本节引言：

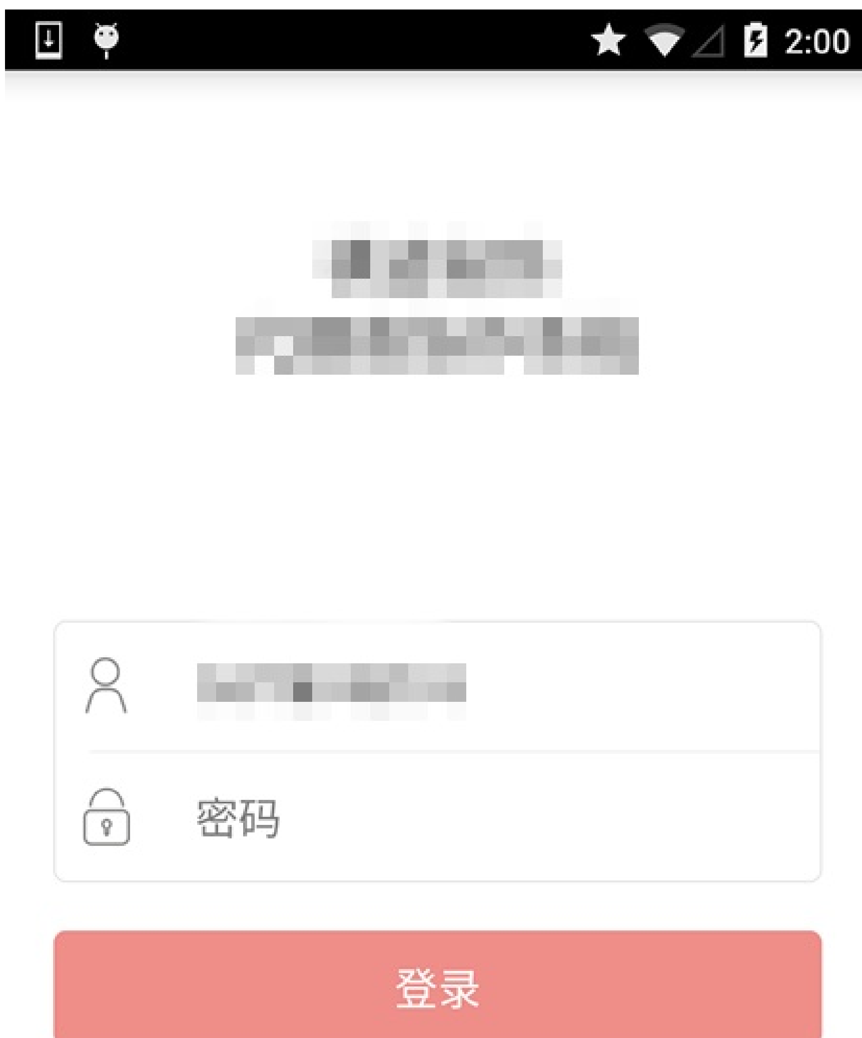
上一节中我们学习了第一个 UI控件**TextView**（文本框），文中给出了很多实际开发中可能遇到的一些需求 的解决方法，应该会为你的开发带来便利，在本节中，我们来学习第二个很常用的控件**EditText(输入框)**；和TextView非常类似，最大的区别是：EditText可以接受用户输入！和前面一样，我们不一个个讲属性，只讲实际应用，要扣属性可以自己查看API文档：API文档；那么开始本节内容！

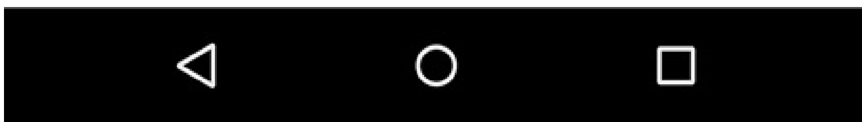
1.设置默认提示文本

如下图，相信你对于这种用户登录的界面并不陌生，是吧，我们很多时候都用的这种界面



相比另外这种，下面这种又如何？





还不赖是吧，当然，不会在这里贴布局，这里只介绍默认提示文本的两个控制属性：

默认提示文本的两个属性如下：

```
android:hint="默认提示文本"
android:textColorHint="#95A1AA"
```

前者设置提示的文本内容，后者设置提示文本的颜色！

2.获得焦点后全选组件内所有文本内容

当我们点击想当我们的输入框获得焦点后，不是将光标移动到文本的开始或者结尾；而是 获取到输入框中所有的文本内容的话！这个时候我们可以使用**selectAllOnFocus**属性

```
android:selectAllOnFocus="true"
```

比如下面的效果图: 第一个是设置了该属性的，第二个是没设置该属性的，设置为true的EditText获得焦点后 选中的是所有文本！



3.限制EditText输入类型

有时我们可能需要对输入的数据进行限制，比如输入电话号码的时候，你输入了一串字母，这 显然是不符合我们预期的，而限制输入类型可以通过inputType属性来实现！

比如限制只能为电话号码,密码(**textPassword**)：

```
<EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" />
```

可选参数如下：

文本类型，多为大写、小写和数字符号

```
android:inputType="none"
android:inputType="text"
android:inputType="textCapCharacters"
android:inputType="textCapWords"
android:inputType="textCapSentences"
android:inputType="textAutoCorrect"
android:inputType="textAutoComplete"
android:inputType="textMultiLine"
android:inputType="textImeMultiLine"
android:inputType="textNoSuggestions"
android:inputType="textUri"
android:inputType="textEmailAddress"
android:inputType="textEmailSubject"
android:inputType="textShortMessage"
android:inputType="textLongMessage"
android:inputType="textPersonName"
android:inputType="textPostalAddress"
android:inputType="textPassword"
android:inputType="textVisiblePassword"
android:inputType="textWebEditText"
android:inputType="textFilter"
android:inputType="textPhonetic"
```

数值类型

```
android:inputType="number"
android:inputType="numberSigned"
android:inputType="numberDecimal"
android:inputType="phone"//拨号键盘
android:inputType="datetime"
android:inputType="date"//日期键盘
android:inputType="time"//时间键盘
```

4.设置最小行，最多行，单行，多行，自动换行

EditText默认是多行显示的，并且能够自动换行，即当一行显示不完的时候，他会自动换到第二行

> 我们可以对其进行限制，比如 设置最小行的行数：**android:minLines="3"** 或者设置EditText最大的行数：**android:maxLines="3"** PS:当输入内容超过maxline,文字会自动向上滚动！！

另外很多时候我们可能要限制EditText只允许单行输入，而且不会滚动，比如上面的登陆界面的 例子，我们只需要设置

```
android:singleLine="true"</p>
<p>
即可实现单行输入不换行</p>

<hr />

<h2>5.设置文字间隔，设置英文字母大写类型</h2>

<p>我们可以通过下述两个属性来设置字的间距：
</p>
<pre>
android:textScaleX="1.5"    //设置字与字的水平间隔
android:textScaleY="1.5"    //设置字与字的垂直间隔
```

另外EditText还为我们提供了设置英文字母大写类型的属性：**android:capitalize** 默认none，提供了三个可选值：

> - **sentences**：仅第一个字母大写
> - **words**：每一个单词首字母大小，用空格区分单词
> - **characters:**每一个英文字母都大写

6.控制EditText四周的间隔距离与内部文字与边框间的距离

> 我们使用**margin**相关属性增加组件相对其他控件的距离，比如android:marginTop = "5dp" 使用**padding**增加组件内文字和组件边框的距离，比如android:paddingTop = "5dp"

7.设置EditText获得焦点，同时弹出小键盘

> 关于这个EditText获得焦点，弹出小键盘的问题，前不久的项目中纠结了笔者一段时间 需求是：进入Activity后，让EditText获得焦点，同时弹出小键盘供用户输入！ 试了很多网上的方法都不可以，不知道是不是因为笔者用的5.1的系统的问题！ 下面小结下：

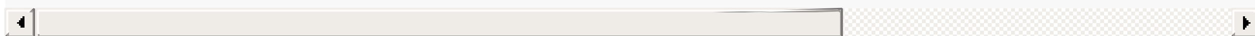
首先是让EditText获得焦点与清除焦点的

> edit.**requestFocus**(); //请求获取焦点 edit.**clearFocus**(); //清除焦点

获得焦点后，弹出小键盘，笔者大部分时间就花在这个上：

> - 低版本的系统直接requestFocus就会自动弹出小键盘了
> - 稍微高一点的版本则需要我们手动地去弹键盘： 第一种：

```
InputMethodManager imm = (InputMethodManager) getSystemService(Cont
imm.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
```



第二种：

```
InputMethodManager imm = (InputMethodManager) getSystemService(Cont
imm.hideSoftInputFromWindow(view.getWindowToken(), 0); //强制隐藏键盘
```



不知道是什么原因，上面这两种方法并没有弹出小键盘，笔者最后使用了:**windowSoftInputMode**属性解决了弹出小键盘的问题，这里跟大家分享下：

> **android:windowSoftInputMode** Activity主窗口与软键盘的交互模式，可以用来避免输入法面板遮挡问题，Android1.5后的一个新特性。 这个属性能影响两件事情：【一】当有焦点产生时，软键盘是隐藏还是显示 【二】是否减少活动主窗口大小以便腾出空间放软键盘
>
> 简单点就是有焦点时的键盘控制以及是否减少Act的窗口大小，用来放小键盘 有下述值可供选择，可设置多个值，用"|"分开 **stateUnspecified**：软键盘的状态并没有指定，系统将选择一个合适的状态或依赖于主题的设置 **stateUnchanged**：当这个activity出现时，软键盘将一直保持在上一个activity里的状态，无论是隐藏还是显示 **stateHidden**：用户选择activity时，软键盘总是被隐藏 **stateAlwaysHidden**：当该Activity主窗口获取焦点时，软键盘也总是被隐藏的 **stateVisible**：软键盘通常是可见的 **stateAlwaysVisible**：用户选择activity时，软键盘总是显示的状态 **adjustUnspecified**：默认设置，通常由系统自行决定是隐藏还是显示 **adjustResize**：该Activity总是调整屏幕的大小以便留出软键盘的空间 **adjustPan**：当前窗口的内容将自动移动以便当前焦点从不被键盘覆盖和用户能总是看到输入内容的部分

我们可以在AndroidManifest.xml为需要弹出小键盘的Activity设置这个属性，比如：

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:windowSoftInputMode="stateVisible" >
```

然后在EditText对象requestFocus()就可以了~

8.EditText光标位置的控制

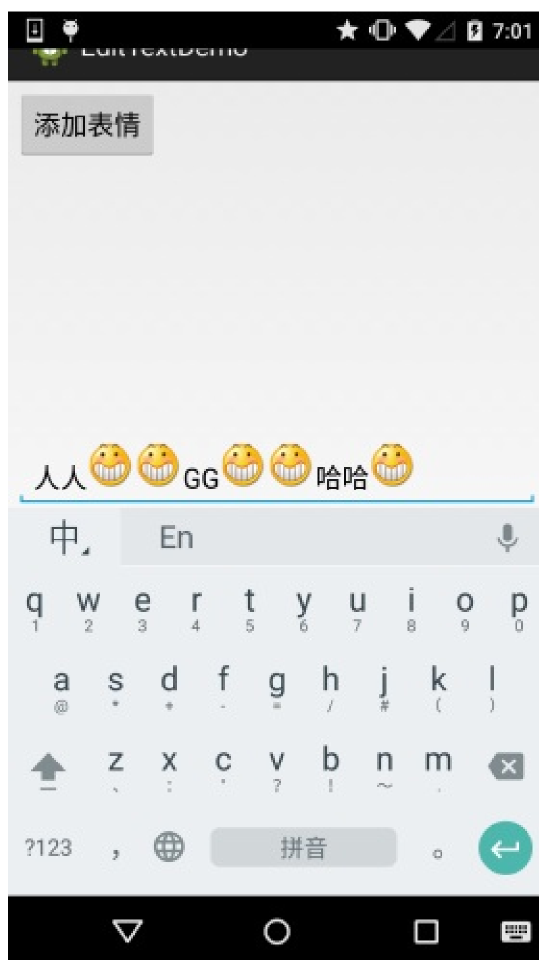
> 有时可能需要我们控制EditText中的光标移动到指定位置或者选中某些文本！EditText为我们提供了**setSelection**()的方法，方法有两种形式:
>
> ```
> setSelection(int index) : void - EditText
> setSelection(int start, int stop) : void - EditText
> ```
>
> 一个参数的是设置光标位置的，两个参数的是设置起始位置与结束位置的中间括的部分，即部分选中！当然我们也可以调用**setSelectAllOnFocus(true)**;让EditText获得焦点时选中全部文本！ 另外我们还可以调用**setCursorVisible(false)**;设置光标不显示 还可以调用**getSelectionStart()**和**getSelectionEnd**获得当前光标的前后位置

9.带表情的EditText的简单实现

相信大家对于QQ或者微信很熟悉吧，我们发送文本的时候可以连同表情一起发送，有两种简单的实现方式：

> 1.使用SpannableString来实现 2.使用Html类来实现 这里笔者用的是第一种，这里只实现一个简单的效果，大家可以把方法抽取出来，自定义一个EditText；也可以自己动手写个类似于QQ那样有多个表情选择的输入框！

看下效果图(点击添加表情即可完成表情添加)：



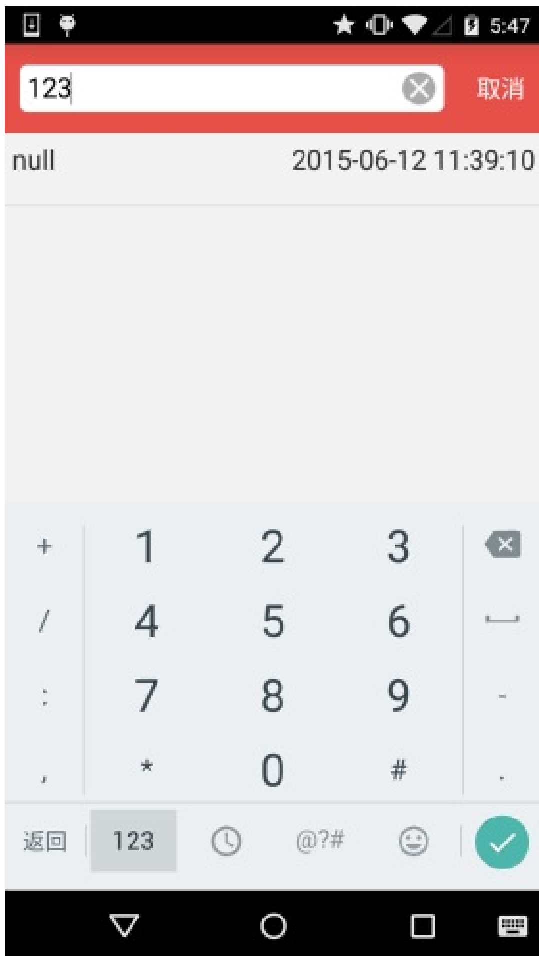
代码也很简单：

```
public class MainActivity extends Activity {
    private Button btn_add;
    private EditText edit_one;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn_add = (Button) findViewById(R.id.btn_add);
        edit_one = (EditText) findViewById(R.id.edit_one);
        btn_add.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                SpannableString spanStr = new SpannableString("imge
                Drawable drawable = MainActivity.this.getResources(
                drawable.setBounds(0,0,drawable.getIntrinsicWidth()
                ImageSpan span = new ImageSpan(drawable,ImageSpan.A
                spanStr.setSpan(span,0,4,Spannable.SPAN_EXCLUSIVE_E
                int cursor = edit_one.getSelectionStart();
                edit_one.getText().insert(cursor, spanStr);
            }
        });
    }
}
```

PS：对了，别忘了放一个图片哦~

10.带删除按钮的EditText

我们常常在App的输入界面上看到：



当我们输入内容后，右面会出现这样一个小叉叉的图标，我们点击后会清空输入框中的内容！实现起来其实也很简单：为EditText设置addTextChangedListener，然后重写TextWatcher（）里的抽象方法，这个用于监听输入框变化的；然后setCompoundDrawablesWithIntrinsicBounds设置小叉叉的图片；最后，重写onTouchEvent方法，如果点击区域是小叉叉图片的位置，清空文本！

实现代码如下：

```
public class EditTextWithDel extends EditText {

    private final static String TAG = "EditTextWithDel";
    private Drawable imgInable;
    private Drawable imgAble;
    private Context mContext;

    public EditTextWithDel(Context context) {
        super(context);
        mContext = context;
        init();
    }

    public EditTextWithDel(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        init();
    }
```

```
    public EditTextWithDel(Context context, AttributeSet attrs, int
        super(context, attrs, defStyleAttr);
        mContext = context;
        init();
    }

    private void init() {
        imgInable = mContext.getResources().getDrawable(R.drawable.
        addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start, in
            }

            @Override
            public void beforeTextChanged(CharSequence s, int start
            }

            @Override
            public void afterTextChanged(Editable s) {
                setDrawable();
            }
        });
        setDrawable();
    }

    // 设置删除图片
    private void setDrawable() {
        if (length() < 1)
            setCompoundDrawablesWithIntrinsicBounds(null, null, nul
        else
            setCompoundDrawablesWithIntrinsicBounds(null, null, img
    }

    // 处理删除事件
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (imgAble != null && event.getAction() == MotionEvent.AC
            int eventX = (int) event.getRawX();
            int eventY = (int) event.getRawY();
            Log.e(TAG, "eventX = " + eventX + "; eventY = " + event
            Rect rect = new Rect();
            getGlobalVisibleRect(rect);
            rect.left = rect.right - 100;
            if (rect.contains(eventX, eventY))
                setText("");
        }
        return super.onTouchEvent(event);
    }
    @Override
    protected void finalize() throws Throwable {
        super.finalize();
    }
```

```
}
```

本节小结：

> 本节给大家介绍了Android UI控件中的EditText(输入框)控件，用法有很多，当然上述情况肯定满足不了实际需求的，实际开发中我们可能需要根据自己的需求来自定义EditText！当然，这就涉及到了自定义控件这个高级一点的主题了，在进阶部分我们会对Android中的自定义控件进行详细的讲解！现在会用就可以了~

2.3.3 Button(按钮)与ImageButton(图像按钮)

本节引言：

> 今天给大家介绍的Android基本控件中的两个按钮控件，Button普通按钮和ImageButton图像按钮；其实ImageButton和Button的用法基本类似，至于与图片相关的则和后面ImageView相同，所以本节 只对Button进行讲解，另外Button是TextView的子类，所以TextView上很多属性也可以应用到Button 上！我们实际开发中对于Button的，无非是对按钮的几个状态做相应的操作，比如：按钮按下的时候 用一种颜色，弹起又一种颜色，或者按钮不可用的时候一种颜色这样！上述实现无非是通过 **StateListDrawable**这种Drawable资源来实现，即编写一个drawable的资源文件，就说这么多， 直接开始本节内容~

1.StateListDrawable简介：

> StateListDrawable是Drawable资源的一种，可以根据不同的状态，设置不同的图片效果，关键节点 **< selecotr >**，我们只需要讲Button的blackground属性设置为该drawable资源即可轻松实现，按下 按钮时不同的按钮颜色或背景！

我们可以设置的属性：

> - **drawable**:引用的Drawable位图,我们可以把他放到最前面,就表示组件的正常状态~
> - **state_focused**:是否获得焦点
> - **state_window_focused**:是否获得窗口焦点
> - **state_enabled**:控件是否可用
> - **state_checkable**:控件可否被勾选,eg:checkbox
> - **state_checked**:控件是否被勾选
> - **state_selected**:控件是否被选择,针对有滚轮的情况
> - **state_pressed**:控件是否被按下
> - **state_active**:控件是否处于活动状态,eg:slidingTab
> - **state_single**:控件包含多个子控件时,确定是否只显示一个子控件
> - **state_first**:控件包含多个子控件时,确定第一个子控件是否处于显示状态
> - **state_middle**:控件包含多个子控件时,确定中间一个子控件是否处于显示状态
> - **state_last**:控件包含多个子控件时,确定最后一个子控件是否处于显示状态

2.实现按钮的按下效果：

好的，先准备三个图片背景，一般我们为了避免按钮拉伸变形都会使用.9.png作为按钮的drawable！ 先来看下 运行效果图：



代码实现：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_pressed="true" android:drawable="@drawable/
    <item android:state_enabled="false" android:drawable="@drawable
    <item android:drawable="@drawable/ic_course_bg_cheng"/>
</selector>
```

布局文件:**activity_main.xml**

```
<Button
        android:id="@+id/btnOne"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:background="@drawable/btn_bg1"
        android:text="按钮"/>

    <Button
        android:id="@+id/btnTwo"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:text="按钮不可用"/>
```

MainActivity.java：

```
public class MainActivity extends Activity {

    private Button btnOne,btnTwo;

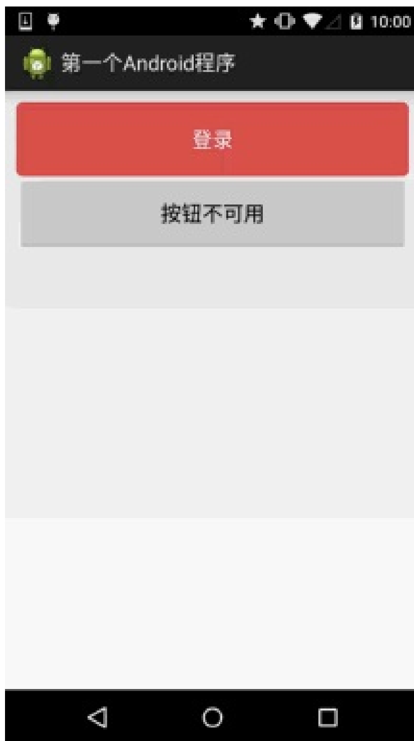
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnOne = (Button) findViewById(R.id.btnOne);
        btnTwo = (Button) findViewById(R.id.btnTwo);
        btnTwo.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if(btnTwo.getText().toString().equals("按钮不可用"))
                    btnOne.setEnabled(false);
                    btnTwo.setText("按钮可用");
                }else{
                    btnOne.setEnabled(true);
                    btnTwo.setText("按钮不可用");
                }
            }
        });
    }

}
```

3.使用颜色值绘制圆角按钮

> 很多时候我们不一定会有美工是吧，或者我们不会PS或毁图秀秀，又或者我们懒，不想自己去做图， 这个时候我们可以自己写代码来作为按钮背景，想要什么颜色就什么颜色，下面我们来定制个圆角的 的按钮背景~，这里涉及到另一个drawable资源:ShapeDrawable，这里不详细讲，后面会详细介绍每一个drawable~这里会用就好，只是EditText修改下Background属性而已，这里只贴drawable资源！

先看下效果图：



bbuton_danger_rounded.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic

    <item android:state_pressed="true"><shape>
            <solid android:color="@color/bbutton_danger_pressed" /:
            <stroke android:width="1dp" android:color="@color/bbutt
            <corners android:radius="@dimen/bbuton_rounded_corner_r
        </shape></item>

    <item android:state_enabled="false"><shape>
        <solid android:color="@color/bbutton_danger_disabled" />
            <stroke android:width="1dp" android:color="@color/bbutt
            <corners android:radius="@dimen/bbuton_rounded_corner_r
        </shape></item>

    <item><shape>
            <solid android:color="@color/bbutton_danger" />
            <stroke android:width="1dp" android:color="@color/bbutt
            <corners android:radius="@dimen/bbuton_rounded_corner_r
        </shape></item>

</selector>
```

color.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="bbutton_danger_pressed">#ffd2322d</color>
    <color name="bbutton_danger_edge">#ffd43f3a</color>
    <color name="bbutton_danger_disabled">#a5d9534f</color>
    <color name="bbutton_danger_disabled_edge">#a5d43f3a</color>
    <color name="bbutton_danger">#ffd9534f</color>
    <color name="text_font_white">#FFFFFF</color>
</resources>
```

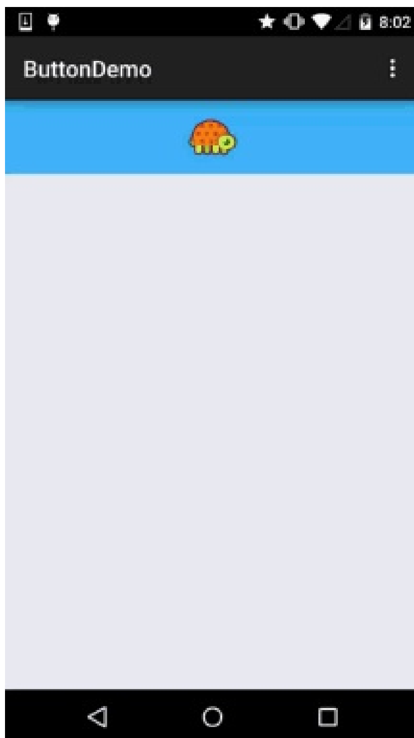
dimens.xml:

```
<dimen name="bbuton_rounded_corner_radius">5dp</dimen>
```

4.实现Material Design水波效果的Button

如果你的Android手机是5.0以上的系统，相信对下面这种按钮点击效果并不会陌生：

实现效果图：



快的那个是按下后的效果，慢的是长按后的效果！

实现逻辑：

1.我们继承ImageButton，当然你可以换成Button或者View,这里笔者想把龟放到中间才继承ImageButton

2.首先，创建两个Paint(画笔)对象，一个绘制底部背景颜色，一个绘制波纹扩散的

3.接着计算最大半径，开始半径每隔一段时间递增一次，直到等于最大半径，然后重置状态！

PS:大概的核心，刚学可能对自定义View感到陌生，没事，这里了解下即可，以后我们会讲，当然 你可以自己扣扣，注释还是蛮详细的~

实现代码：

自定义ImageButton：**MyButton.java**

```
package demo.com.jay.buttondemo;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.os.SystemClock;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.ViewConfiguration;
import android.widget.ImageButton;

/**
 * Created by coder-pig on 2015/7/16 0016.
 */
public class MyButton extends ImageButton {

    private static final int INVALIDATE_DURATION = 15;     //每次刷
    private static int DIFFUSE_GAP = 10;                    //扩散半径
    private static int TAP_TIMEOUT;                          //判断点

    private int viewWidth, viewHeight;                     //控件宽高
    private int pointX, pointY;                            //控件原点
    private int maxRadio;                               //扩散的最大半
    private int shaderRadio;                          //扩散的半径

    private Paint bottomPaint, colorPaint;            //画笔:背景和水波
    private boolean isPushButton;                   //记录是否按钮被按下

    private int eventX, eventY;                   //触摸位置的X,Y坐标
    private long downTime = 0;                  //按下的时间

    public MyButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        initPaint();
        TAP_TIMEOUT = ViewConfiguration.getLongPressTimeout();
    }

    /*
    * 初始化画笔
    * */
    private void initPaint() {
```

```
        colorPaint = new Paint();
        bottomPaint = new Paint();
        colorPaint.setColor(getResources().getColor(R.color.reveal_
        bottomPaint.setColor(getResources().getColor(R.color.bottor
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                if (downTime == 0) downTime = SystemClock.elapsedRe
                eventX = (int) event.getX();
                eventY = (int) event.getY();
                //计算最大半径：
                countMaxRadio();
                isPushButton = true;
                postInvalidateDelayed(INVALIDATE_DURATION);
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                if(SystemClock.elapsedRealtime() - downTime < TAP_T
                    DIFFUSE_GAP = 30;
                    postInvalidate();
                }else{
                    clearData();
                }
                break;
        }
        return super.onTouchEvent(event);
    }

    @Override
    protected void dispatchDraw(Canvas canvas) {
        super.dispatchDraw(canvas);
        if(!isPushButton) return; //如果按钮没有被按下则返回
        //绘制按下后的整个背景
        canvas.drawRect(pointX, pointY, pointX + viewWidth, pointY
        canvas.save();
        //绘制扩散圆形背景
        canvas.clipRect(pointX, pointY, pointX + viewWidth, pointY
        canvas.drawCircle(eventX, eventY, shaderRadio, colorPaint);
        canvas.restore();
        //直到半径等于最大半径
        if(shaderRadio < maxRadio){
            postInvalidateDelayed(INVALIDATE_DURATION,
                    pointX, pointY, pointX + viewWidth, pointY + vi
            shaderRadio += DIFFUSE_GAP;
        }else{
            clearData();
        }
    }

    /*
```

```
         * 计算最大半径的方法
         * */
    private void countMaxRadio() {
        if (viewWidth > viewHeight) {
            if (eventX < viewWidth / 2) {
                maxRadio = viewWidth - eventX;
            } else {
                maxRadio = viewWidth / 2 + eventX;
            }
        } else {
            if (eventY < viewHeight / 2) {
                maxRadio = viewHeight - eventY;
            } else {
                maxRadio = viewHeight / 2 + eventY;
            }
        }
    }

    /*
    * 重置数据的方法
    * */
    private void clearData(){
        downTime = 0;
        DIFFUSE_GAP = 10;
        isPushButton = false;
        shaderRadio = 0;
        postInvalidate();
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh)
        super.onSizeChanged(w, h, oldw, oldh);
        this.viewWidth = w;
        this.viewHeight = h;
    }
}
```

```
<code>
<p><b>color.xml：</b></p>
<pre>
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="reveal_color">#FFFFFF</color>
    <color name="bottom_color">#3086E4</color>
    <color name="bottom_bg">#40BAF8</color>
</resources>
```

activity_main.xml：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <demo.com.jay.buttondemo.MyButton
        android:id="@+id/myBtn"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:src="@mipmap/ic_tur_icon"
        android:background="@color/bottom_bg"
        android:scaleType="center"/>

</RelativeLayout>
```

源码下载(AS工程的哦)：ButtonDemo.zip

本节小结：

本节给大家介绍了Button在实际开发中的一些用法，可能有些东西我们还没学，这里 知道下即可，后面学到自然会深入讲解，谢谢~

2.3.4 ImageView(图像视图)

本节引言：

本节介绍的UI基础控件是：ImageView(图像视图)，见名知意，就是用来显示图像的一个View或者说控件！ 官方API:ImageView;本节讲解的内容如下：

1. ImageView的src属性和blackground的区别；
2. adjustViewBounds设置图像缩放时是否按长宽比
3. scaleType设置缩放类型
4. 最简单的绘制圆形的ImageView

1.src属性和background属性的区别：

> 在API文档中我们发现ImageView有两个可以设置图片的属性，分别是：src和background
>
> 常识：
>
> ①background通常指的都是背景,而src指的是内容!!
>
> ②当使用**src**填入图片时,是按照图片大小直接填充,并不会进行拉伸
>
> 而使用background填入图片,则是会根据ImageView给定的宽度来进行拉伸

1)写代码验证区别：

写个简单的布局测试下：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

效果图如下：

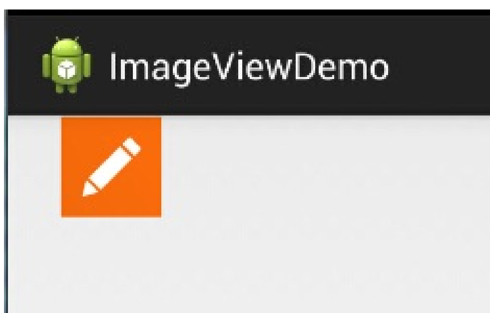


结果分析：

宽高都是wrap_content那就一样,是原图大小,但是,当我们固定了宽或者高的话, 差别就显而易见了,blackground完全填充了整个ImageView,而src依旧是那么大, 而且他居中了哦,这就涉及到了ImageView的另一个属性scaleType了! 另外还有一点,这里我们说了只设置width或者height哦！加入我们同时设置了 width和height的话,blackground依旧填充,但是,src的大小可能发生改变哦! 比如,我们测试下下面这段代码:

```
<ImageView  android:layout_width="100dp"  android:layout_height="5(
```

运行效果图：



PS:scaleType下面会讲~

2)解决blackground拉伸导致图片变形的方法

在前面的效果图中的第二个Imageview中我们可以看到图片已经被拉伸变形了, 正方形变成了长方形,对于和我一样有轻微强迫症的人来说,显然是不可接受的, 有没有办法去设置呢?答案肯定是有的,笔者暂时知道的有以下两种方式:

- 这个适用于动态加载ImageView的,代码也渐渐,只要在添加View的时候,把大小写死就可以了

```
LinearLayout.LayoutParams layoutParam =  new  LinearLayout.Layo
```

- 除了动态加载view,更多的时候,我们还是会通过xml布局的方式引入ImageView的 解决方法也不难,就是通过drawable的Bitmap资源文件来完成,然后blackground属性设置为该文件即可! 这个xml文件在drawable文件夹下创建,这个文件夹是要自己创建的哦!！

pen_bg.xml:

```
<bitmap  xmlns:android="http://schemas.android.com/apk/res/android"
```

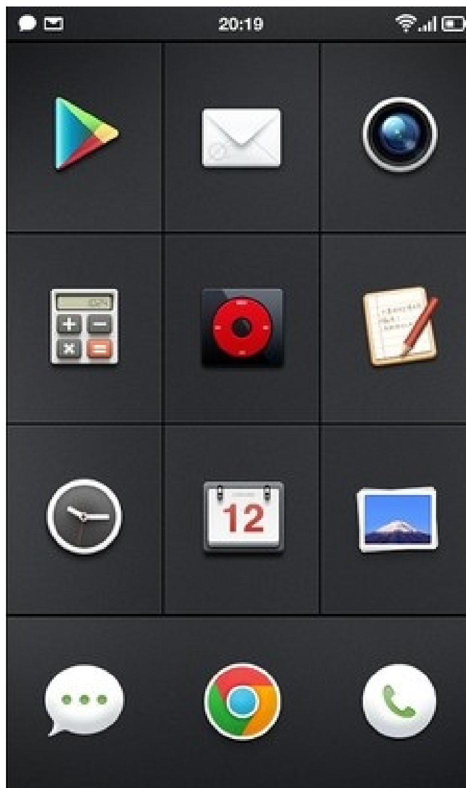
> 上述代码并不难理解,估计大家最迷惑的是titleMode属性吧,这个属性是平铺,就是我们windows设置 背景时候的平铺,多个小图标铺满整个屏幕捏！记得了吧！不记得自己可以试试!disabled就是把他给禁止了!
>
> 就是上面这串简单的代码,至于调用方法如下:
>
> 动态: ibtnPen.setBacklgroundResource(R.drawable.penbg);
>
> 静态: android:background = "@drawable/penbg"

3)设置透明度的问题

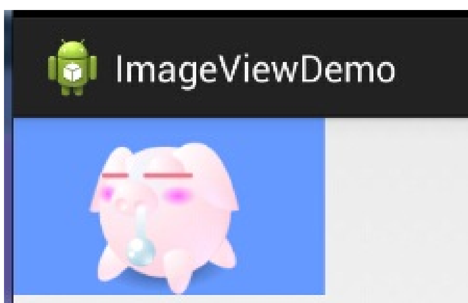
> 说完前面两个区别,接着再说下setAlpha属性咯!这个很简单,这个属性,只有**src**时才是有效果的**!!**

4)两者结合妙用:

网上的一张图：



一看去是一个简单的GridView,每个item都是一个ImageView,但是细心的你可能发现了, 上面的ICON都不是规则的,而是圆形,圆角矩形等等,于是乎这里用到了src + background了！ 要实现上述的效果,你只需要两个操作: 找一张透明的**png**图片 **+** 设置一个黑色的背景 (当然你也可以设置png的透明度来实现,不过结果可能和预想的有出入哦!) 我们写个简单例子：



如图， 呆萌呆萌的小猪就这样显示到ImageView上了,哈哈,blackground设置了蓝色背景!

实现代码：

```
<ImageView  android:layout_width="150dp"  android:layout_height="wr
```

PS: 当然你也可以用selctor实现点击效果,设置不同的情况设置不同的图片,以实现点击或者触摸效果!

5)Java代码中设置blackground和src属性:

> 前景(对应src属性):**setImageDrawable**( ); 背景(对应background属性):**setBackgroundDrawable**( );

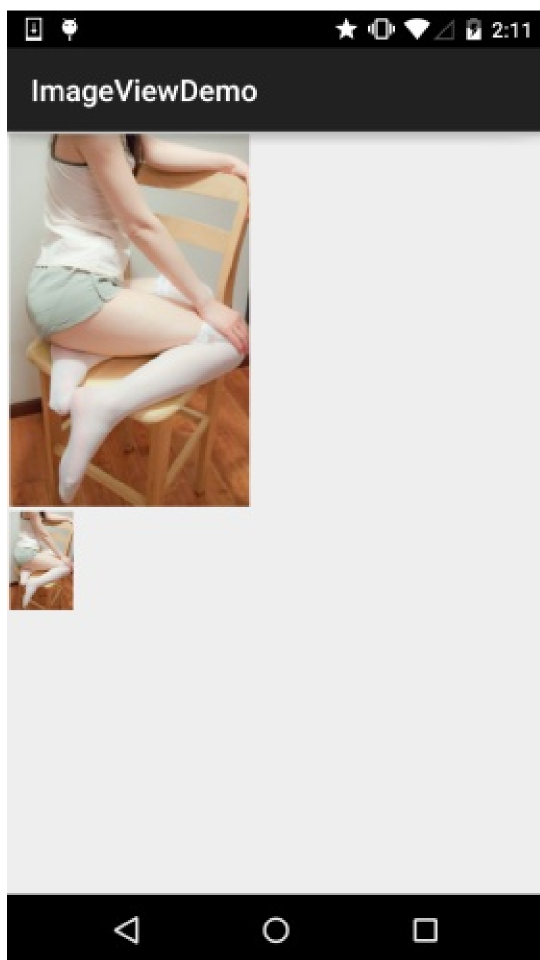
2.adjustViewBounds设置缩放是否保存原图长宽比

> ImageView为我们提供了**adjustViewBounds**属性，用于设置缩放时是否保持原图长宽比！ 单独设置不起作用，需要配合**maxWidth**和**maxHeight**属性一起使用！而后面这两个属性 也是需要adjustViewBounds为true才会生效的~
>
> - android:maxHeight:设置ImageView的最大高度
> - android:maxWidth:设置ImageView的最大宽度

代码示例：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

运行效果图：



结果分析： 大的那个图片是没有任何处理的图片,尺寸是:541_374;而下面的那个的话我们通过maxWidth和maxHeight 限制ImageView最大宽度与高度为200px，就是最多只能显示200_200的图片,我们又设置了一个 **adjustViewBounds = "true"**调整我们的边界来保持图片的长宽比,此时的ImageView宽高为是128*200~

3.scaleType设置缩放类型

> android:scaleType用于设置显示的图片如何缩放或者移动以适应ImageView的大小 Java代码中可以通过imageView.setScaleType(ImageView.ScaleType.CENTER);来设置~ 可选值如下：
>
> - **fitXY**:对图像的横向与纵向进行独立缩放,使得该图片完全适应ImageView,但是图片的横纵比可能会发生改变
> - **fitStart**:保持纵横比缩放图片,知道较长的边与Image的编程相等,缩放完成后将图片放在ImageView的左上角
> - **fitCenter**:同上,缩放后放于中间;
> - **fitEnd**:同上,缩放后放于右下角;
> - **center**:保持原图的大小， 显示在ImageView的中心。当原图的size大于ImageView的size，超过部分裁剪处理。
> - **centerCrop**:保持横纵比缩放图片,知道完全覆盖ImageView,可能会出现图片的显示不完全
> - **centerInside**:保持横纵比缩放图片,直到ImageView能够完全地显示图片
> - **matrix**:默认值， 不改变原图的大小， 从ImageView的左上角开始绘制原图， 原图超过ImageView的部分作裁剪处理

接下来我们一组组的来对比：

1)1.fitEnd,fitStart,fitCenter

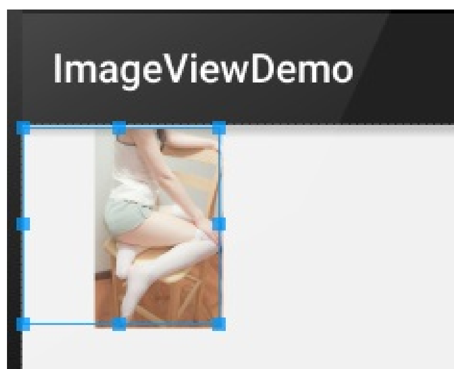
这里以fitEnd为例，其他两个类似：

示例代码：

```
<!-- 保持图片的横纵比缩放,知道该图片能够显示在ImageView组件上,并将缩放好的图
```



运行效果图：



2)centerCrop与centerInside

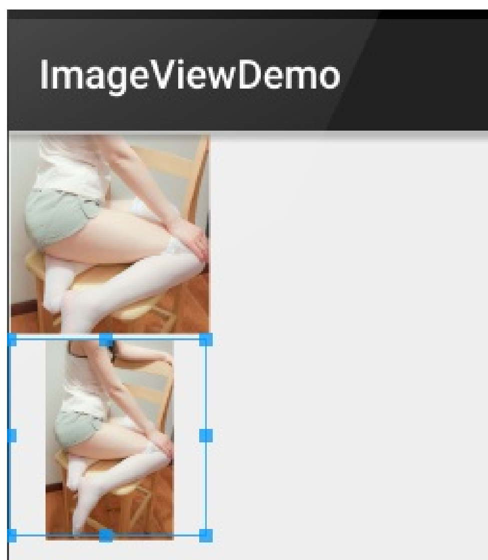
- centerCrop:按横纵比缩放,直接完全覆盖整个ImageView

- centerInside:按横纵比缩放,使得ImageView能够完全显示这个图片

示例代码：

```
<ImageView  android:layout_width="300px"  android:layout_height="30
```

运行效果图：



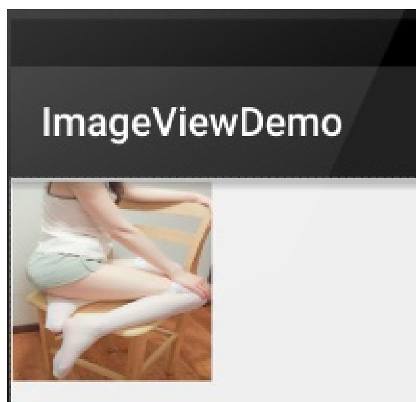
3)fitXY

不按比例缩放图片，目标是把图片塞满整个View

示例代码：

```
<ImageView  android:layout_width="300px"  android:layout_height="30
```

运行效果图：



好吧，明显扁了=-=~

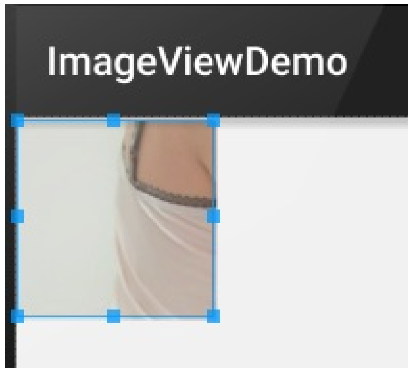
4)matrix

从ImageView的左上角开始绘制原图，原图超过ImageView的部分作裁剪处理

示例代码：

```
<ImageView  android:layout_width="300px"  android:layout_height="3(
```

运行效果图：



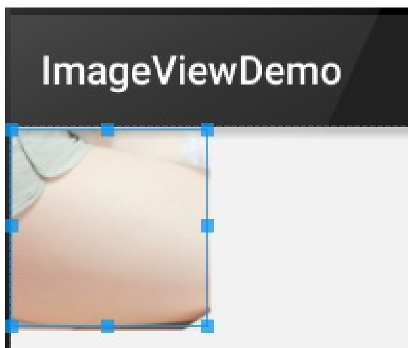
5)center

保持原图的大小，显示在ImageView的中心。当原图的size大于ImageView的size，超过部分裁剪处理。

示例代码：

```
<ImageView  android:layout_width="300px"  android:layout_height="3(
```

运行效果图：



4.最简单的绘制圆形的ImageView

相信大家对圆形或者圆角的ImageView不陌生吧，现在很多的APP都很喜欢圆形的头像是吧~

这里就简单的写个圆形的ImageView吧，当然这只是一个示例，再不考虑性能与抗锯齿的情况下！！！

可以说是写来玩玩，实际项目的话可以考虑用Github上牛人写的控件，比如下面这两个：

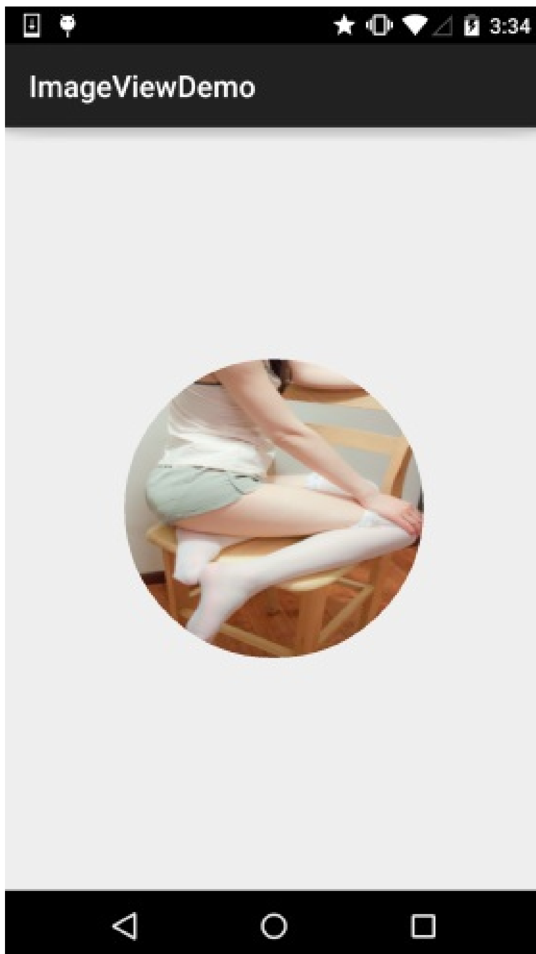
git怎么玩前面已经教过大家了~把项目clone下来把相关文件复制到自己的项目即可~

RoundedImageView

CircleImageView

代码示例：

运行效果图：



实现代码：

自定义ImageView：**RoundImageView.java

```
package com.jay.demo.imageviewdemo;  import android.content.Context
 * Created by coder-pig on 2015/7/18 0018.
 */  public  class  RoundImageView  extends  ImageView  {  private
```

布局代码：**activity_main.xml:**

```
<com.jay.demo.imageviewdemo.RoundImageView  android:id="@+id/img_ro
```

MainActivity.java:

```
package com.jay.demo.imageviewdemo;  import android.graphics.Bitmap
```

本节小结：

> 本节讲解了ImageView(图像视图),内容看上很多，不过都是一些详述性的东西，知道即可~ 最后的自定义圆形ImageView也是，只是写来玩玩的，实际项目中还是建议使用那两个第三方的自定义控件吧~

2.3.5.RadioButton(单选按钮)&Checkbox(复选框)

本节引言：

> 本节给大家带来的是Andoird基本UI控件中的RadioButton和Checkbox; 先说下本节要讲解的内容是：RadioButton和Checkbox的 **1.**基本用法 **2.**事件处理； **3.**自定义点击效果； **4.**改变文字与选择框的相对位置； **5.**修改文字与选择框的距离

其实这两个控件有很多地方都是类似的，除了单选和多选，事件处理，其他的都是类似的！ 另外还有一个ListView上Checkbox的错位的问题，我们会在ListView那一章对这个问题进行 解决，好的，开始本节内容~ 本节官方文档API：RadioButton；CheckBox；

1.基本用法与事件处理：

1）RadioButton(单选按钮)

> 如题单选按钮，就是只能够选中一个，所以我们需要把RadioButton放到RadioGroup按钮组中，从而实现 单选功能！先熟悉下如何使用RadioButton，一个简单的性别选择的例子： 另外我们可以为外层RadioGroup设置orientation属性然后设置RadioButton的排列方式，是竖直还是水平~

效果图：



PS:笔者的手机是Android 5.0.1的，这里的RadioButton相比起旧版本的RadioButton，稍微好看一点~

布局代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请选择性别"
        android:textSize="23dp"
        />

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/btnMan"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男"
            android:checked="true"/>

        <RadioButton
            android:id="@+id/btnWoman"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女"/>
    </RadioGroup>

    <Button
        android:id="@+id/btnpost"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="提交"/>

</LinearLayout>
```

获得选中的值：

这里有两种方法,

第一种是为RadioButton设置一个事件监听器setOnCheckChangeListener

例子代码如下：

```
RadioGroup radgroup = (RadioGroup) findViewById(R.id.radioGroup);
        //第一种获得单选按钮值的方法
        //为radioGroup设置一个监听器:setOnCheckedChanged()
        radgroup.setOnCheckedChangeListener(new OnCheckedChangeList
            @Override
            public void onCheckedChanged(RadioGroup group, int chec
                RadioButton radbtn = (RadioButton) findViewById(che
                Toast.makeText(getApplicationContext(), "按钮组值发生
            }
        });
```



运行效果图：

按钮组值发生改变,你选了男

按钮组值发生改变,你选了女

PS：另外有一点要切记，要为每个RadioButton添加一个id，不然单选功能会生效！！！

第二种方法是通过单击其他按钮获取选中单选按钮的值，当然我们也可以直接获取，这个看需求~

例子代码如下：

```
Button btnchange = (Button) findViewById(R.id.btnpost);
        RadioGroup radgroup = (RadioGroup) findViewById(R.id.radioG
        //为radioGroup设置一个监听器:setOnCheckedChanged()
        btnchange.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                for (int i = 0; i < radgroup.getChildCount(); i++)
                    RadioButton rd = (RadioButton) radgroup.getChil
                    if (rd.isChecked()) {
                        Toast.makeText(getApplicationContext(), "点
                        break;
                    }
                }
            }
        });
```



运行效果图：

点击提交按钮,获取你选择的是:女

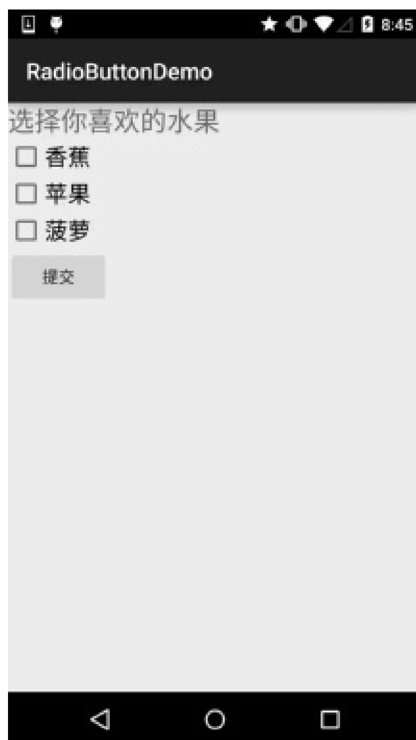
代码解析：这里我们为提交按钮设置了一个setOnClickListener事件监听器,每次点击的话遍历一次RadioGroup判断哪个按钮被选中我们可以通过下述方法获得RadioButton的相关信息！

- **getChildCont**( )获得按钮组中的单选按钮的数目；
- **getChinIdAt**(i):根据索引值获取我们的单选按钮
- **isChecked**( ):判断按钮是否选中

2）CheckBox(复选框)

> 如题复选框，即可以同时选中多个选项，至于获得选中的值，同样有两种方式： 1.为每个CheckBox添加事件：setOnCheckedChangeListener 2.弄一个按钮，在点击后，对每个checkbox进行判断:isChecked()；

运行效果图：



实现代码：

```
public class MainActivity extends AppCompatActivity implements View

    private CheckBox cb_one;
    private CheckBox cb_two;
    private CheckBox cb_three;
    private Button btn_send;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        cb_one = (CheckBox) findViewById(R.id.cb_one);
        cb_two = (CheckBox) findViewById(R.id.cb_two);
        cb_three = (CheckBox) findViewById(R.id.cb_three);
        btn_send = (Button) findViewById(R.id.btn_send);

        cb_one.setOnCheckedChangeListener(this);
        cb_two.setOnCheckedChangeListener(this);
        cb_three.setOnCheckedChangeListener(this);
        btn_send.setOnClickListener(this);

    }

    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boo
       if(compoundButton.isChecked()) Toast.makeText(this,compoundB
    }

    @Override
    public void onClick(View view) {
        String choose = "";
        if(cb_one.isChecked())choose += cb_one.getText().toString(
        if(cb_two.isChecked())choose += cb_two.getText().toString(
        if(cb_three.isChecked())choose += cb_three.getText().toStri
        Toast.makeText(this,choose,Toast.LENGTH_SHORT).show();
    }
}
```

2.自定义点击效果

> 虽然5.0后的RadioButton和Checkbox都比旧版本稍微好看了点，但是对于我们来说 可能还是不喜欢或者需求，需要自己点击效果！实现起来很简单，先编写一个自定义 的selctor资源，设置选中与没选中时的切换图片~！

实现效果图如下：



PS:这里素材的原因，有点小...

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item
        android:state_enabled="true"
        android:state_checked="true"
        android:drawable="@mipmap/ic_checkbox_checked"/>
    <item
        android:state_enabled="true"
        android:state_checked="false"
        android:drawable="@mipmap/ic_checkbox_normal" />
</selector>
```

写好后，我们有两种方法设置，也可以说一种吧！你看看就知道了~

①android:button属性设置为上述的selctor

```
android:button="@drawable/rad_btn_selctor"
```

②在style中定义一个属性，然后通过android style属性设置，先往style添加下述代码：

```
    <style name="MyCheckBox" parent="@android:style/Widget.Compounc
        <item name="android:button">@drawable/rad_btn_selctor</iter
    </style>
```

然后布局那里:

```
style="@style/MyCheckBox"
```

3.改变文字与选择框的相对位置

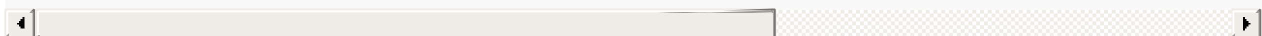
> 这个实现起来也很简单，还记得我们之前学TextView的时候用到的drawableXxx吗？ 要控制选择框的位置，两部即可！设置：
>
> **Step 1.** android:button="@null" **Step 2.** android:drawableTop="@android:drawable/btn_radio" 当然我们可以把drawableXxx替换成自己喜欢的效果！

4.修改文字与选择框的距离

> 有时，我们可能需要调节文字与选择框之间的距离，让他们看起来稍微没那么挤，我们可以： 1.在XML代码中控制： 使用android:paddingXxx = "xxx" 来控制距离 2.在Java代码中，稍微好一点，动态计算paddingLeft!

示例代码如下：

```
rb.setButtonDrawable(R.drawable.rad_btn_selctor);
int rb_paddingLeft = getResources().getDrawable(R.mipmap.ic_checkbo
rb.setPadding(rb_paddingLeft, 0, 0, 0);
```

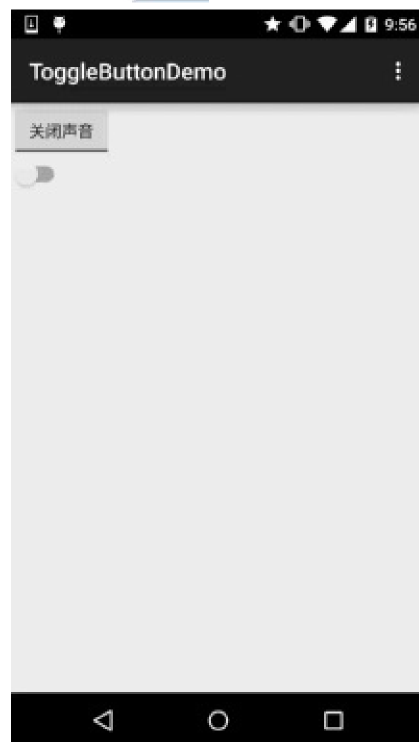


本节小结：

> 好的，关于RadioButton和Checkbox就讲到这里，如果有什么写得不对的，不好的，或者有好的建议欢迎指出 万分感激~谢谢...

2.3.6 开关按钮ToggleButton和开关Switch

本节引言：

> 本节给大家介绍的Android基本UI控件是:开关按钮ToggleButton和开关Switch,可能大家对着两个组件 并不熟悉，突然想起笔者的第一间外包公司，是否在wifi下联网的开关，竟然用的TextView，然后叫美工 且两个切换前后的图，然后代码中进行设置，当然点击TextView的时候判断状态，然后设置对应的背景...
>
> 好吧，也是醉了，好吧...本节讲解的两个其实都是开关组件，只是后者需要在Android 4.0以后才能使用 所以AndroidManifest.xml文件中的minsdk需要 >= 14 否则会报错~，先来看看这两个控件长什么样先， Android 5.0后这两个控件相比以前来说好看了许多，先看下5.0前的样子：
>
> **5.0以前的ToggleButton和Switch：** 
>
> 
>
> **5.0**版本： 
>
>
> 好吧，鲜明的对比...接下来我们就来学习者两个控件的使用吧，其实两个的使用几乎是相同的。
>
> 开始之前贴下官方API先：Switch；ToggleButton

1.核心属性讲解：

1）ToggleButton(开关按钮)

可供我们设置的属性：

> - **android:disabledAlpha**：设置按钮在禁用时的透明度
> - **android:textOff**：按钮没有被选中时显示的文字
> - **android:textOn**：按钮被选中时显示的文字 另外，除了这个我们还可以自己写个selector，然后设置下Background属性即可~

2) Switch(开关)

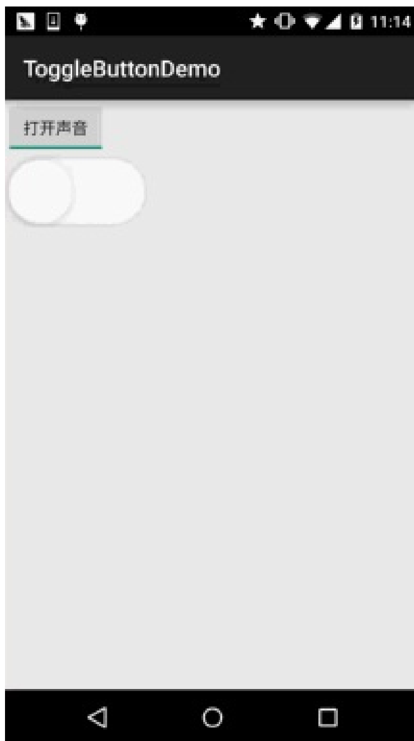
可供我们设置的属性：

> - **android:showText**：设置on/off的时候是否显示文字,boolean
> - **android:splitTrack**：是否设置一个间隙，让滑块与底部图片分隔,boolean
> - **android:switchMinWidth**：设置开关的最小宽度
> - **android:switchPadding**：设置滑块内文字的间隔
> - **android:switchTextAppearance**：设置开关的文字外观，暂时没发现有什么用...
> - **android:textOff**：按钮没有被选中时显示的文字
> - **android:textOn**：按钮被选中时显示的文字
> - **android:textStyle**：文字风格，粗体，斜体写划线那些
> - **android:track**：底部的图片
> - **android:thumb**：滑块的图片
> - **android:typeface**：设置字体，默认支持这三种:sans, serif, monospace; 除此以外还可以使用 其他字体文件(***.ttf**)，首先要将字体文件保存在assets/fonts/目录下，不过需要在Java代码中设置： **Typeface typeFace =Typeface.createFromAsset(getAssets(),"fonts/HandmadeTypewriter.ttf"); textView.setTypeface(typeFace);**

2.使用示例：

因为比较简单，所以我们把他们写到一起，另外，我们为Switch设置下滑块和底部的图片，实现 一个类似于IOS 7的滑块的效果，但是有个缺点就是不能在XML中对滑块和底部的大小进行设置， 就是素材多大，Switch就会多大，我们可以在Java中获得Drawable对象，然后对大小进行修改， 简单的例子：

运行效果图：



实现代码： 先是两个drawable的文件： **thumb_selctor.xml:**

```
<?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="h
```

track_selctor.xml:

```
<?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="h
```

布局文件**:activity_main.xml:**

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/an
```

MainActivity.java：

```
public  class  MainActivity  extends  AppCompatActivity  implements
```

2.3.7 ProgressBar(进度条)

本节引言：

本节给大家带来的是Android基本UI控件中的ProgressBar(进度条)，ProgressBar的应用场景很多，比如 用户登录时，后台在发请求，以及等待服务器返回信息，这个时候会用到进度条；或者当在进行一些比较 耗时的操作，需要等待一段较长的时间，这个时候如果没有提示，用户可能会以为程序Carsh或者手机死机 了，这样会大大降低用户体验，所以在需要进行耗时操作的地方，添加上进度条，让用户知道当前的程序 在执行中，也可以直观的告诉用户当前任务的执行进度等！使用进度条可以给我带来这样的便利！ 好了，开始讲解本节内容~ 对了，ProgressBar官方API文档：ProgressBar

1.常用属性讲解与基础实例

从官方文档，我们看到了这样一个类关系图：



ProgressBar继承与View类， 直接子类有AbsSeekBar和ContentLoadingProgressBar， 其中AbsSeekBar的子类有SeekBar和RatingBar，可见这二者也是基于ProgressBar实现的

常用属性详解：

> - android:**max**：进度条的最大值
> - android:**progress**：进度条已完成进度值
> - android:**progressDrawable**：设置轨道对应的Drawable对象
> - android:**indeterminate**：如果设置成true，则进度条不精确显示进度
> - android:**indeterminateDrawable**：设置不显示进度的进度条的Drawable对象
> - android:**indeterminateDuration**：设置不精确显示进度的持续时间
> - android:**secondaryProgress**：二级进度条，类似于视频播放的一条是当前播放进度，一条是缓冲进度，前者通过progress属性进行设置！

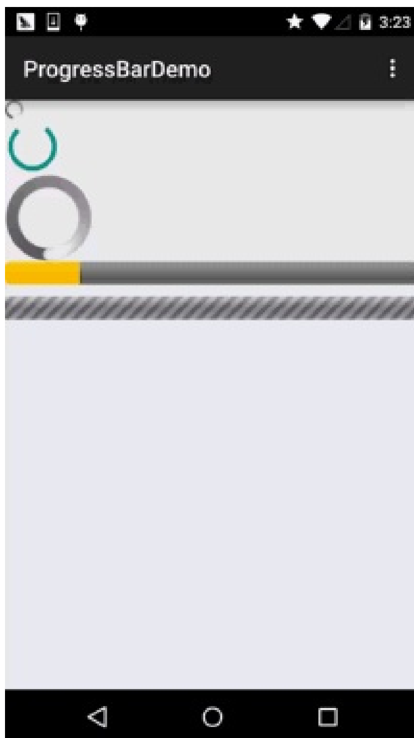
对应的再Java中我们可调用下述方法：

> - **getMax**()：返回这个进度条的范围的上限
> - **getProgress**()：返回进度
> - **getSecondaryProgress**()：返回次要进度
> - **incrementProgressBy**(int diff)：指定增加的进度
> - **isIndeterminate**()：指示进度条是否在不确定模式下
> - **setIndeterminate**(boolean indeterminate)：设置不确定模式下

接下来来看看系统提供的默认的进度条的例子吧！

系统默认进度条使用实例：

运行效果图：



实现布局代码：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

好吧，除了第二个能看，其他的就算了...系统提供的肯定是满足不了我们的需求的！ 下面我们就来讲解下实际开发中我们对进度条的处理！

2.使用动画来替代圆形进度条

第一个方案是，使用一套连续图片，形成一个帧动画，当需要进度图的时候，让动画可见，不需要 的时候让动画不可见即可！而这个动画，一般是使用AnimationDrawable来实现的！好的，我们来 定义一个AnimationDrawable文件：

PS:用到的图片素材：进度条图片素材打包.zip

运行效果图：



实现步骤：

在res目录下新建一个:anim文件件，然后创建amin_pgbar.xml的资源文件：

```
<?xml version="1.0" encoding="utf-8"?> <animation-list xmlns:andr
```

接着写个布局文件，里面仅仅有一个ImageView即可，用于显示进度条，把src设置为上述drawable资源即可！ 最后到MainActivity.java

```
public class MainActivity extends AppCompatActivity { private
```

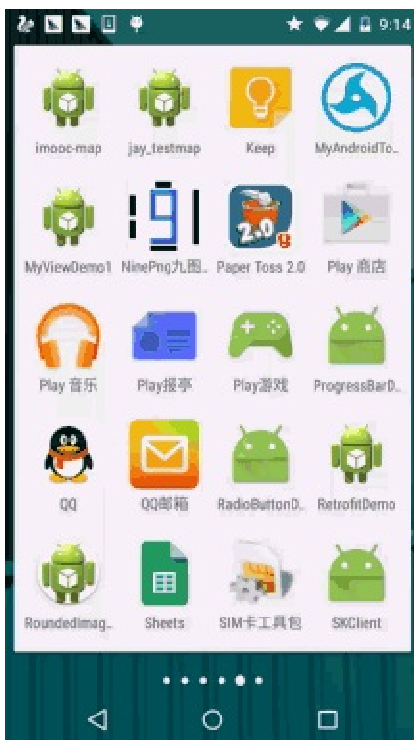
这里只是写了如何启动动画，剩下的就你自己来了哦~在需要显示进度条的时候，让ImageView可见；在不需要的时候让他隐藏即可！另外其实Progressbar本身有一个indeterminateDrawable，只需把 这个参数设置成上述的动画资源即可，但是

进度条条的图案大小是不能直接修改的，需要Java代码中 修改，如果你设置了宽高，而且这个宽高过大的时候，你会看到有多个进度条...自己权衡下吧~

3.自定义圆形进度条

> 相信你看完2会吐槽，卧槽，这么坑爹，拿个动画来坑人，哈哈，实际开发中都这样，当然上述 这种情况只适用于不用显示进度的场合，如果要显示进度的场合就没用处了，好吧，接下来看下 网上一个简单的自定义圆形进度条！代码还是比较简单，容易理解，又兴趣可以看看，或者进行相关扩展~

运行效果图：



实现代码：

自定义**View**类：

```
/**
 * Created by Jay on 2015/8/5 0005.
 */  public  class  CirclePgBar  extends  View  {  private  Paint r
```

然后在布局文件中加上：

```
<com.jay.progressbardemo.CirclePgBar  android:layout_width="match
```

就是这么简单~

本节小结：

本节给大家介绍了Android中的常用控件：ProgressBar讲解了基本用法，以及实际开发中 对于进度条的两种实现方法，第二个自定义进度条可以自行完善，然后用到实际开发中~！ 好的，本节就到这里，谢谢~

2.3.8 SeekBar(拖动条)

本节引言：

> 本节我们继续来学习Android的基本UI控件中的拖动条——SeekBar，相信大家对他并不陌生，最常见的 地方就是音乐播放器或者视频播放器了，音量控制或者播放进度控制，都用到了这个SeekBar，我们 先来看看SeekBar的类结构，来到官方文档：SeekBar
>
> java.lang.Object
> ↳android.view.View
> ↳android.widget.ProgressBar
> ↳android.widget.AbsSeekBar
> ↳android.widget.SeekBar
>
> 嘿嘿，这玩意是ProgressBar的子类耶，也就是ProgressBar的属性都可以用咯！ 而且他还有一个自己的属性就是：**android:thumb**，就是允许我们自定义滑块~ 好的，开始本节内容！

1.SeekBar基本用法

好吧，基本用法其实很简单，常用的属性无非就下面这几个常用的属性，Java代码里只要setXxx即可：

> **android:max**="100" //滑动条的最大值
>
> **android:progress**="60" //滑动条的当前值
>
> **android:secondaryProgress**="70" //二级滑动条的进度
>
> **android:thumb** = "@mipmap/sb_icon" //滑块的drawable

接着要说下SeekBar的事件了，**SeekBar.OnSeekBarChangeListener** 我们只需重写三个对应的方法：

> **onProgressChanged**：进度发生改变时会触发
>
> **onStartTrackingTouch**：按住SeekBar时会触发
>
> **onStopTrackingTouch**：放开SeekBar时触发

简单的代码示例:

效果图:



实现代码：

```
public class MainActivity extends AppCompatActivity {

    private SeekBar sb_normal;
    private TextView txt_cur;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        bindViews();
    }

    private void bindViews() {
        sb_normal = (SeekBar) findViewById(R.id.sb_normal);
        txt_cur = (TextView) findViewById(R.id.txt_cur);
        sb_normal.setOnSeekBarChangeListener(new SeekBar.OnSeekBarC
            @Override
            public void onProgressChanged(SeekBar seekBar, int prog
                txt_cur.setText("当前进度值:" + progress + "  / 100 
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
                Toast.makeText(mContext, "触碰SeekBar", Toast.LENGT
            }

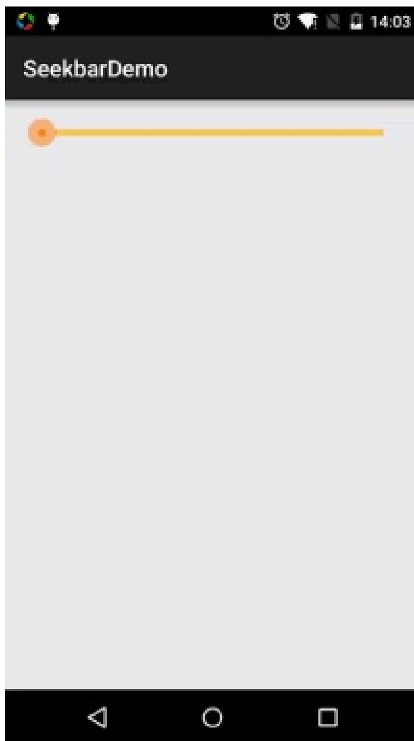
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                Toast.makeText(mContext, "放开SeekBar", Toast.LENGT
            }
        });
    }
}
```

2.简单SeekBar定制：

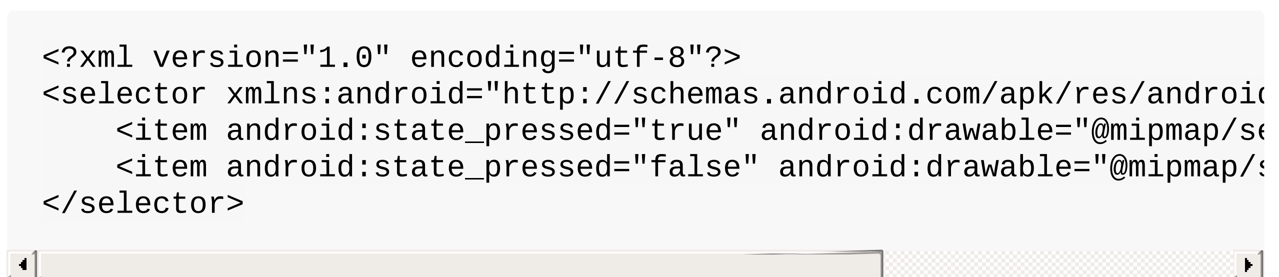
> 本来还想着自定义下SeekBar的，后来想想，还是算了，涉及到自定义View的一些东西，可能初学者并 不了解，看起来也有点难度，关于自定义View的还是放到进阶那里吧，所以这里就只是简单的定制下SeekBar！ 定制的内容包括滑块，以及轨道！

代码实例：

运行效果图：

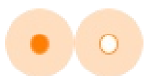


代码实现： **1.滑块状态Drawable：sb_thumb.xml**



```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_pressed="true" android:drawable="@mipmap/se
    <item android:state_pressed="false" android:drawable="@mipmap/s
</selector>
```

贴下素材：



2.条形栏Bar的Drawable：sb_bar.xml

这里用到一个layer-list的drawable资源！其实就是层叠图片，依次是:背景，二级进度条，当前进度：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <solid android:color="#FFFFD042" />
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip>
            <shape>
                <solid android:color="#FFFFFFFF" />
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip>
            <shape>
                <solid android:color="#FF96E85D" />
            </shape>
        </clip>
    </item>
</layer-list>
```

3.然后布局引入**SeekBar**后，设置下**progressDrawable**与**thumb**即可！

```
<SeekBar
        android:id="@+id/sb_normal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:maxHeight="5.0dp"
        android:minHeight="5.0dp"
        android:progressDrawable="@drawable/sb_bar"
        android:thumb="@drawable/sb_thumb"/>
```

就是这么简单！

本节小结：

好的，关于SeekBar就到这里，谢谢大家~

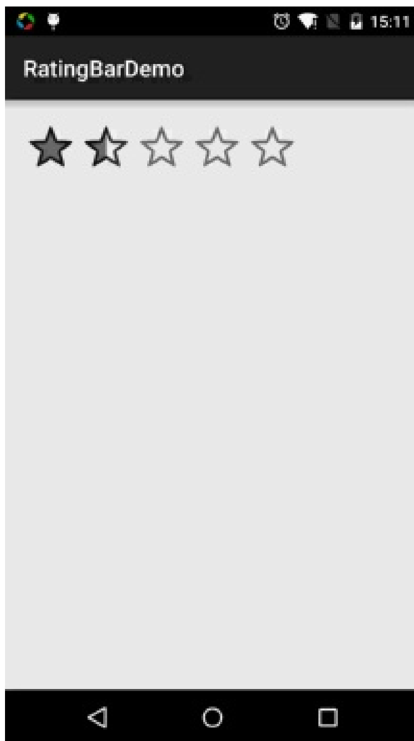
2.3.9 RatingBar(星级评分条)

本节引言：

> 上一节的SeekBar是不是很轻松咧，本节我们学的这个RatingBar(星级评分条)也非常简单，相信在某宝，买过东西的对这个应该不陌生，收到卖家的包裹，里面很多时候会有个小纸片，五星好评返还多少元这样，而评分的时候就可以用到我们这个星级评分条了~先来瞅瞅官方文档 官方文档：RatingBar 我们可以看到，这玩意和SeekBar的类结构是一样的，也是ProgressBar的子类：
>
> public class
>
> RatingBar
>
> extends AbsSeekBar
>
> java.lang.Object
> ↳android.view.View
> ↳android.widget.ProgressBar
> ↳android.widget.AbsSeekBar
> ↳android.widget.RatingBar
>
> ▸ Known Direct Subclasses
>
> AppCompatRatingBar
>
> 也就是说他同样有用ProgressBar的相关属性，接下来我们来探究RatingBar特有的属性！

1.RatingBar基本使用：

先来看看5.0的原生SeekBar长什么样：



——相关属性：

android:isIndicator：是否用作指示，用户无法更改，默认false **android:numStars**：显示多少个星星，必须为整数 **android:rating**：默认评分值，必须为浮点数 **android:stepSize**： 评分每次增加的值，必须为浮点数

除了上面这些，还有两种样式供我们选择咧，但是不建议使用，因为这两种样式都好丑... 他们分别是： **style="?android:attr/ratingBarStyleSmall" style="?android:attr/ratingBarStyleIndicator"**

——事件处理： 只需为RatingBar设置**OnRatingBarChangeListener**事件，然后重写下**onRatingChanged()**方法即可！

实现代码如下：

```
public class MainActivity extends AppCompatActivity { private
```

2.定制环节：

> 嘿嘿，我们很多时候不会用星星作为评分标准的，我们来改改呗~ 把星星改成其他的，比如笑脸，两个素材：



接下来和前面的SeekBar一样编写一个layer-list的文件：

ratingbar_full.xml:

```
<?xml version="1.0" encoding="utf-8"?> <layer-list xmlns:android=
```

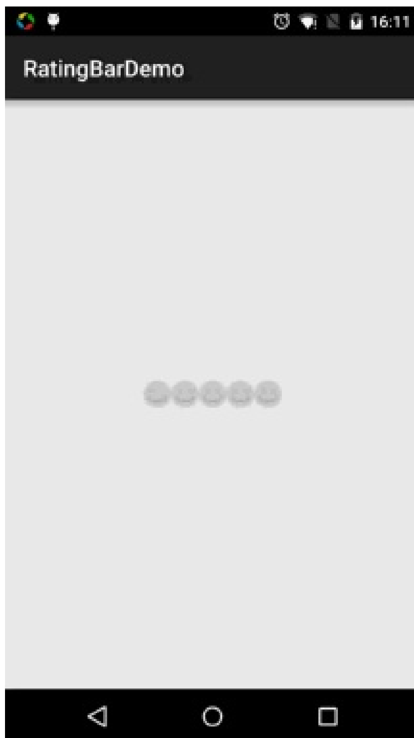
接着在style.xml中自定义下RatingBar Style，在**style.xml**加上这个：

```
<style name="roomRatingBar" parent="@android:style/Widget.RatingB
```

最后在布局中的Ratingbar组件设置下：

```
<RatingBar android:id="@+id/rb_normal" style="@style/roomRatingBa
```

运行效果图：



好的，效果还可以哈，至于间距问题，就需要对图片坐下处理了，就是需要切图的时候在图片左右预留点空格~!

本节小结：

好的，关于RatingBar的使用就到这里，和前面的SeekBar其实大同小异，蛮轻松的~嗯，谢谢~

2.4.1 ScrollView(滚动条)

本节引言：

> 本节带来的是Android基本UI控件中的第十个：**ScrollView**(滚动条)，或者我们应该叫他 竖直滚动条，对应的另外一个水平方向上的滚动条：**HorizontalScrollView**，先来一发官方文档 的链接：ScrollView，我们可以看到类的结构如下：
>
> public class
>
> ScrollView
>
> extends FrameLayout
>
> java.lang.Object
> ↳android.view.View
> ↳android.view.ViewGroup
> ↳android.widget.FrameLayout
> ↳android.widget.ScrollView
>
> 嘿嘿，原来是一个FrameLayout的容器，不过在他的基础上添加了滚动，允许显示的比实际多的内容！
>
> 另外，只能够往里面放置一个子元素，可以是单一的组件，又或者一个布局包裹着的复杂的层次结构！
>
> 一般对于可能显示不完的情况，我们可以直接在布局的外层套上一个: ScrollView或者HorizontalScrollView！就这么简单~！

可能遇到的一些需求

好的，就不一个个扣文档了，直接说实际开发中可能会遇到的一些需求吧：

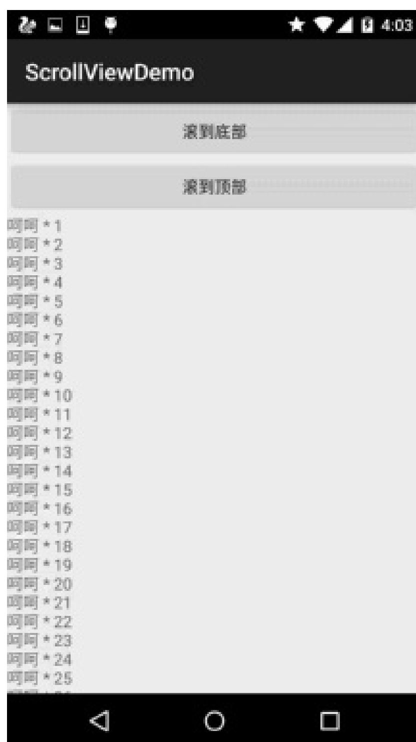
另外有一个很典型的问题就是:ScrollView和ListView的嵌套问题，这个放到ListView那一章节 再来讲解~

1.滚动到底部：

> 我们可以直接利用ScrollView给我们提供的:**fullScroll()**方法：
>
> scrollView.fullScroll(ScrollView.**FOCUS_DOWN**);滚动到底部
>
> scrollView.fullScroll(ScrollView.**FOCUS_UP**);滚动到顶部
>
> 另外用这玩意的时候要小心异步的玩意，就是addView后，有可能还没有显示完，如果这个时候直接调用该方法的话，可能会无效，这就需要自己写handler来更新了~

代码示例：

效果图：



实现代码：

布局比较简单，就不贴了，直接贴MainActivity **MainActivity.java**

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_down;
    private Button btn_up;
    private ScrollView scrollView;
    private TextView txt_show;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_down = (Button) findViewById(R.id.btn_down);
        btn_up = (Button) findViewById(R.id.btn_up);
        scrollView = (ScrollView) findViewById(R.id.scrollView);
        txt_show = (TextView) findViewById(R.id.txt_show);
        btn_down.setOnClickListener(this);
        btn_up.setOnClickListener(this);

        StringBuilder sb = new StringBuilder();
        for (int i = 1; i <= 100; i++) {
            sb.append("呵呵 * " + i + "\n");
        }
        txt_show.setText(sb.toString());

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_down:
                scrollView.fullScroll(ScrollView.FOCUS_DOWN);
                break;
            case R.id.btn_up:
                scrollView.fullScroll(ScrollView.FOCUS_UP);
                break;
        }
    }
}
```

当然除了这种方法还，你还可以使用另一种复杂一点的写法：

```
public static void scrollToBottom(final View scroll, final View inn
    Handler mHandler = new Handler();
    mHandler.post(new Runnable() {
        public void run() {
            if (scroll == null || inner == null) {
                return;
            }
            int offset = inner.getMeasuredHeight() - scroll.getHei
            if (offset < 0) {
                offset = 0;
            }
            scroll.scrollTo(0, offset);
        }
    });
}
```

scrollTo()参数依次为x，y滚到对应的x，y位置！

2.设置滚动的滑块图片

> 这个更加简单： 垂直方向滑块：android:**scrollbarThumbVertical** 水平方向滑块：android:**scrollbarThumbHorizontal**

3.隐藏滑块

好吧，这个好像没什么卵用：

> 方法有两种： 1.android:scrollbars="none" 2.Java代码设置：scrollview.setVerticalScrollBarEnabled(false);

4.设置滚动速度：

> 这个并没有给我们提供可以直接设置的方法，我们需要自己继承ScrollView，然后重写一个 public void fling (int velocityY)的方法：
>
> ```
> @Override
> public void fling(int velocityY) {
>     super.fling(velocityY / 2);    //速度变为原来的一半
> }
> ```

本节小结：

好的，能想到的ScrollView的东西就这么多，因为平时这个用得并不多，一般直接套在外面而已， 另外，问题最多的一般是ScrollView和ListView的嵌套问题~如果有什么补充欢迎提出，谢谢~

2.4.2 Date & Time组件(上)

本节引言：

> 本节给大家带来的是Android给我们提供的显示时间的几个控件，他们分别是：TextClock，AnalogClock，Chronometer，另外其实还有个过时的DigitalClock就不讲解了！好的，开始本节内容！

1.TextClock(文本时钟)

> TextClock是在Android 4.2(API 17)后推出的用来替代DigitalClock的一个控件！TextClock可以以字符串格式显示当前的日期和时间，因此推荐在Android 4.2以后使用TextClock。这个控件推荐在24进制的android系统中使用，TextClock提供了两种不同的格式，一种是在24进制中显示时间和日期，另一种是在12进制中显示时间和日期。大部分人喜欢默认的设置。

可以通过调用：TextClock提供的is24HourModeEnabled()方法来查看，系统是否在使用24进制时间显示! 在24进制模式中：

- 如果没获取时间，首先通过getFormat24Hour()返回值；
- 获取失败则通过getFormat12Hour()获取返回值；
- 以上都获取失败则使用默认；

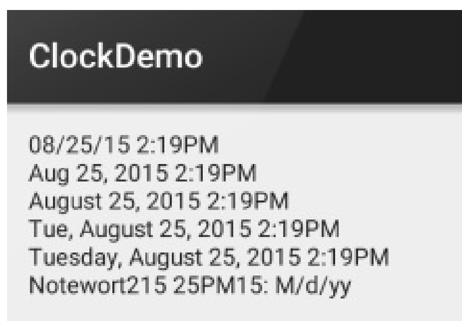
另外他给我们提供了下面这些方法，对应的还有get方法：

Attribute Name	Related Method	Description
android:format12Hour	setFormat12Hour(CharSequence)	设置12时制的格式
android:format24Hour	setFormat24Hour(CharSequence)	设置24时制的格式
android:timeZone	setTimeZone(String)	设置时区

其实更多的时间我们是花在时间形式定义上，就是里面这个CharSequence！这里提供下常用的写法以及结果：

```
<TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MM/dd/yy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="E, MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="EEEE, MMMM dd, yyyy h:mmaa"/>
    <TextClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format12Hour="Noteworthy day: 'M/d/yy"/>
```

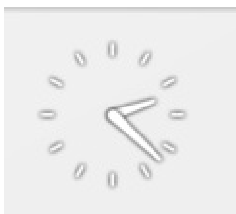
运行结果：



PS: 另外minsdk 要大于或者等于17哦！

2.AnalogClock(模拟时钟)

就是下图这种：



官网中我们可以看到这样三个属性：

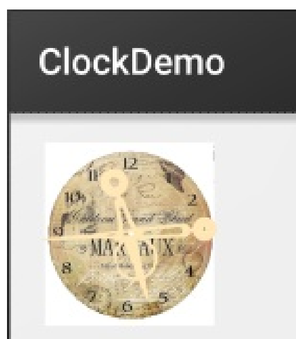
XML Attributes
Attribute Name
android:dial
android:hand_hour
android:hand_minute

依次是：表背景，表时针，分时针的图片，我们可以自行定制：

示例代码如下：

```
<AnalogClock
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:dial="@mipmap/ic_c_bg"
        android:hand_hour="@mipmap/zhen_shi"
        android:hand_minute="@mipmap/zhen_fen" />
```

运行结果：



3.Chronometer(计时器)

如题，就是一个简单的计时器，我们直接上使用示例吧：

使用示例：

实现代码：

布局代码：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Chronometer
        android:id="@+id/chronometer"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#ff0000"
        android:textSize="60dip" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dip"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnStart"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="开始记时" />

        <Button
            android:id="@+id/btnStop"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="停止记时" />

        <Button
            android:id="@+id/btnReset"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="重置" />

        <Button
            android:id="@+id/btn_format"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="格式化" />
    </LinearLayout>

</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View

    private Chronometer chronometer;
    private Button btn_start,btn_stop,btn_base,btn_format;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
    }

    private void initView() {
        chronometer = (Chronometer) findViewById(R.id.chronometer);
        btn_start = (Button) findViewById(R.id.btnStart);
        btn_stop = (Button) findViewById(R.id.btnStop);
        btn_base = (Button) findViewById(R.id.btnReset);
        btn_format = (Button) findViewById(R.id.btn_format);

        chronometer.setOnChronometerTickListener(this);
        btn_start.setOnClickListener(this);
        btn_stop.setOnClickListener(this);
        btn_base.setOnClickListener(this);
        btn_format.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btnStart:
                chronometer.start();// 开始计时
                break;
            case R.id.btnStop:
                chronometer.stop();// 停止计时
                break;
            case R.id.btnReset:
                chronometer.setBase(SystemClock.elapsedRealtime());
                break;
            case R.id.btn_format:
                chronometer.setFormat("Time：%s");// 更改时间显示格式
                break;
        }
    }

    @Override
    public void onChronometerTick(Chronometer chronometer) {
        String time = chronometer.getText().toString();
        if(time.equals("00:00")){
            Toast.makeText(MainActivity.this,"时间到了~",Toast.LENGT
```

```
            }
        }
    }
```

运行截图：



本节小结：

本节跟大家简单的介绍了TextClock，AnalogClock，Chronometer这三个组件，从篇幅就可以看出 其实这几个东西用得并不多，几乎是没用过...知道下就好，用法也超简单... 就这样吧，本节就到这里~谢谢

2.4.3 Date & Time组件(下)

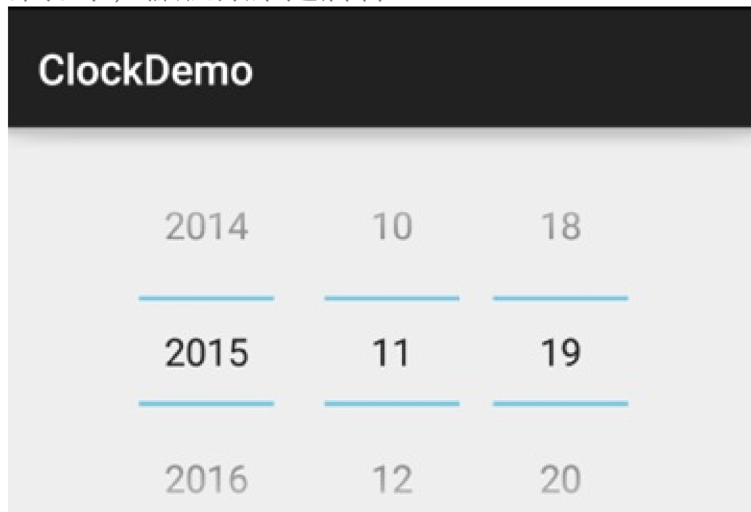
本节引言：

> 本节我们来继续学习Android系统给我们提供的几个原生的Date & Time组件，他们分别是：DatePicker(日期选择器)，TimePicker(时间选择器)，CalendarView(日期视图)，好吧，其实一开始让我扣这几个玩意我是拒绝的，因为在我的印象里，他们是这样的：
>
> 
>
>
> 简直把我丑哭了，有木有，终于知道为什么那么多人喜欢自定义这种类型的控件了！但是毕竟 提纲上写了，自己写的提纲，含着泪也要把他写完...当我把DatePicker写到布局中，然后看下 预览图，哟：

> 
>
>
> 原来，看起来还不错，心情大好，哈哈，那么开始本节内容！

1.DatePicker(日期选择器)

可供我们使用的属性如下：

> - **android:calendarTextColor** ： 日历列表的文本的颜色
> - **android:calendarViewShown**：是否显示日历视图
> - **android:datePickerMode**：组件外观，可选值:spinner，calendar 前者效果如下，默认效果是后者
> 
>
> - **android:dayOfWeekBackground**：顶部星期几的背景颜色
> - **android:dayOfWeekTextAppearance**：顶部星期几的文字颜色
> - **android:endYear**：去年(内容)比如2010
> - **android:firstDayOfWeek**：设置日历列表以星期几开头
> - **android:headerBackground**：整个头部的背景颜色
> - **android:headerDayOfMonthTextAppearance**：头部日期字体的颜色
> - **android:headerMonthTextAppearance**：头部月份的字体颜色
> - **android:headerYearTextAppearance**：头部年的字体颜色
> - **android:maxDate**：最大日期显示在这个日历视图mm / dd / yyyy格式
> - **android:maxDate**：最小日期显示在这个日历视图mm / dd / yyyy格式
> - **android:spinnersShown**：是否显示spinner
> - **android:startYear**：设置第一年(内容)，比如19940年
> - **android:yearListItemTextAppearance**：列表的文本出现在列表中。
> - **android:yearListSelectorColor**：年列表选择的颜色

属性就是上面这些，你想怎么玩就怎么玩，接下来我们说下他的DatePicker的事件： **DatePicker.OnDateChangedListener** 另外，奇怪的是，如果是上面这种mode为calendar的设置了事件并没有响应，看来上面这种 只能选择完后获取对应的值了，如果你的mode未spinner的话，使用下述代码就可以完成事件监听：

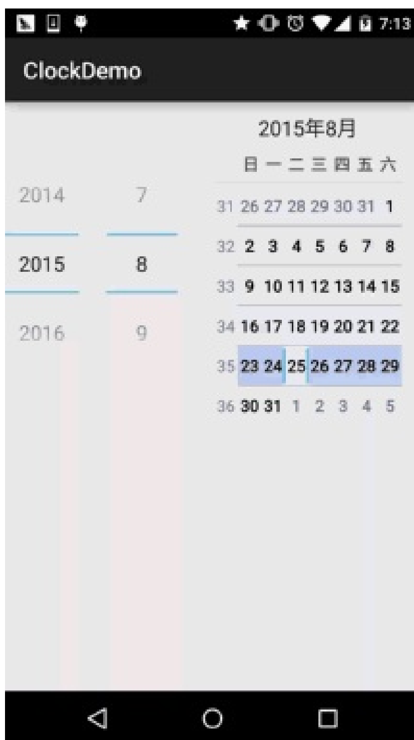
实现代码如下:

```
public class MainActivity extends AppCompatActivity implements Date

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        DatePicker dp_test = (DatePicker) findViewById(R.id.dp_test
        Calendar calendar = Calendar.getInstance();
        int year=calendar.get(Calendar.YEAR);
        int monthOfYear=calendar.get(Calendar.MONTH);
        int dayOfMonth=calendar.get(Calendar.DAY_OF_MONTH);
        dp_test.init(year,monthOfYear,dayOfMonth,this);
    }

    @Override
    public void onDateChanged(DatePicker view, int year, int month
        Toast.makeText(MainActivity.this,"您选择的日期是："+year+"年"
    }
}
```

运行效果图：



2.TimePicker(时间选择器)

> 先来看看5.0的TimePicker长什么样：
>
> 
>
>
> 样子还是蛮标致的哈，我们发现官方给我们提供的属性只有一个：**android:timePickerMode**：组件外观，同样可选值为:spinner和clock(默认) 前者是旧版本的TimePicker~ 而他对应的监听事件是：**TimePicker.OnTimeChangedListener**

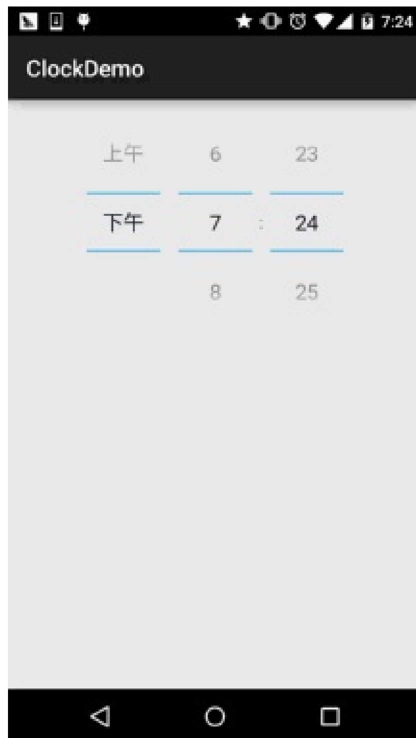
下面来个代码示例：

```
public class MainActivity extends AppCompatActivity{

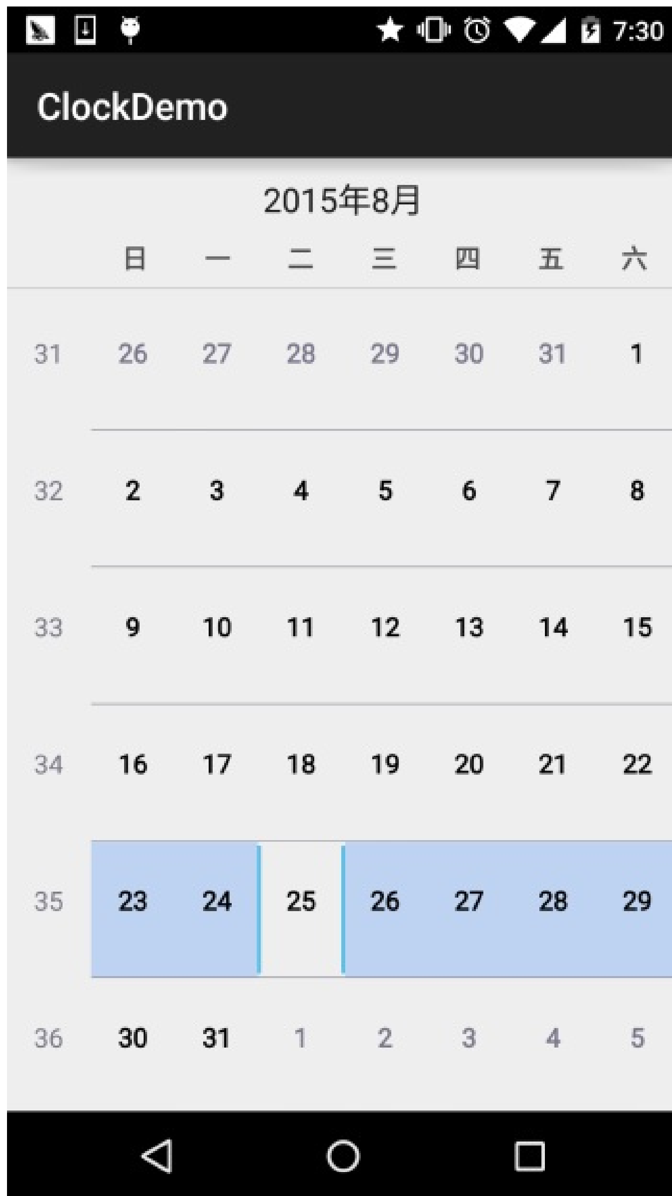
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TimePicker tp_test = (TimePicker) findViewById(R.id.tp_test
        tp_test.setOnTimeChangedListener(new TimePicker.OnTimeChang
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDa
                Toast.makeText(MainActivity.this,"您选择的时间是："+h
            }
        });
    }

}
```

运行效果图： 可惜的是，同样需要旧版本的TimePicker才会触发这个事件！



3.CalendarView(日历视图)

> 好的，一样是看看样子先：
>
> 
>
>
> 嗯，好像变化不大，接下来我们简单的看下文档中给我们提供的属性：

- **android:firstDayOfWeek**：设置一个星期的第一天
- **android:maxDate** ：最大的日期显示在这个日历视图mm / dd / yyyy格式
- **android:minDate**：最小的日期显示在这个日历视图mm / dd / yyyy格式
- **android:weekDayTextAppearance**：工作日的文本出现在日历标题缩写

处理上面的还有其他，但是都是被弃用的... 对应的日期改变事件是：**CalendarView.OnDateChangeListener**

示例代码：

```
public class MainActivity extends AppCompatActivity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        CalendarView cv_test = (CalendarView) findViewById(R.id.cv_
        cv_test.setOnDateChangeListener(new CalendarView.OnDateChar
            @Override
            public void onSelectedDayChange(CalendarView view, int
                Toast.makeText(MainActivity.this,"您选择的时间是："+
            }
        });
    }
}
```

运行效果图：



本节小结：

> 好的，关于这三个控件的介绍就到这里，实际开发中这些控件我们一般都是自定义的，在进阶系列我们会来自己写控件，敬请期待，谢谢~

2.4.4 Adapter基础讲解

本节引言

> 从本节开始我们要讲的UI控件都是跟Adapter(适配器)打交道的，了解并学会使用这个Adapter很重要， Adapter是用来帮助填充数据的中间桥梁， 简单点说就是：将各种数据以合适的形式显示到view上,提供 给用户看！

1.MVC模式的简单理解

> 在开始学习Adapter之前我们要来了解下这个MVC模式概念： 举个例子：大型的商业程序通常由多人一同开发完成,比如有人负责操作接口的规划与设计, 有人负责程序代码的编写如果要能够做到程序项目的分工就必须在程序的结构上做适合的安排 ,如果,接口设计与修改都涉及到程序代码的改变的话,那么两者的分工就会造成执行上的困难 良好的程序架构师将整个程序项目划分为如图的三个部分:

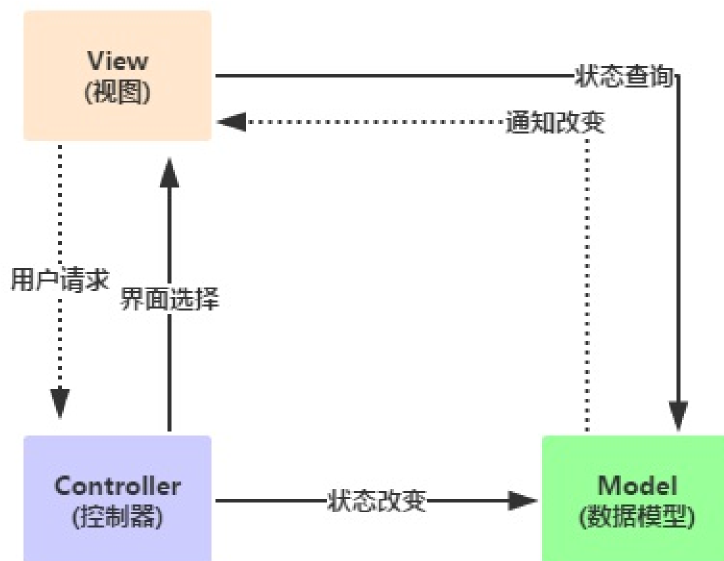


图1:MVC组件类型的关系

关系图解析：

> - **Model**：通常可以理解为数据,负责执行程序的核心运算与判断逻辑,,通过view获得用户输入的数据,然后根据从数据库查询相关的信息,最后进行运算和判断,再将得到的结果交给view来显示
> - **view**:用户的操作接口,说白了就是GUI,应该使用哪种接口组件,组件间的排列位置与顺序都需要设计
> - **Controller**:控制器,作为model与view之间的枢纽,负责控制程序的执行流程以及对象之间的一个互动

而这个Adapter则是中间的这个Controller的部分： **Model**(数据) ---> **Controller**(以什么方式显示到)---> **View**(用户界面) 这就是简单MVC组件的简单理解！

2.Adapter概念解析

官方文档：Adapter

首先我们来看看他的继承结构图：

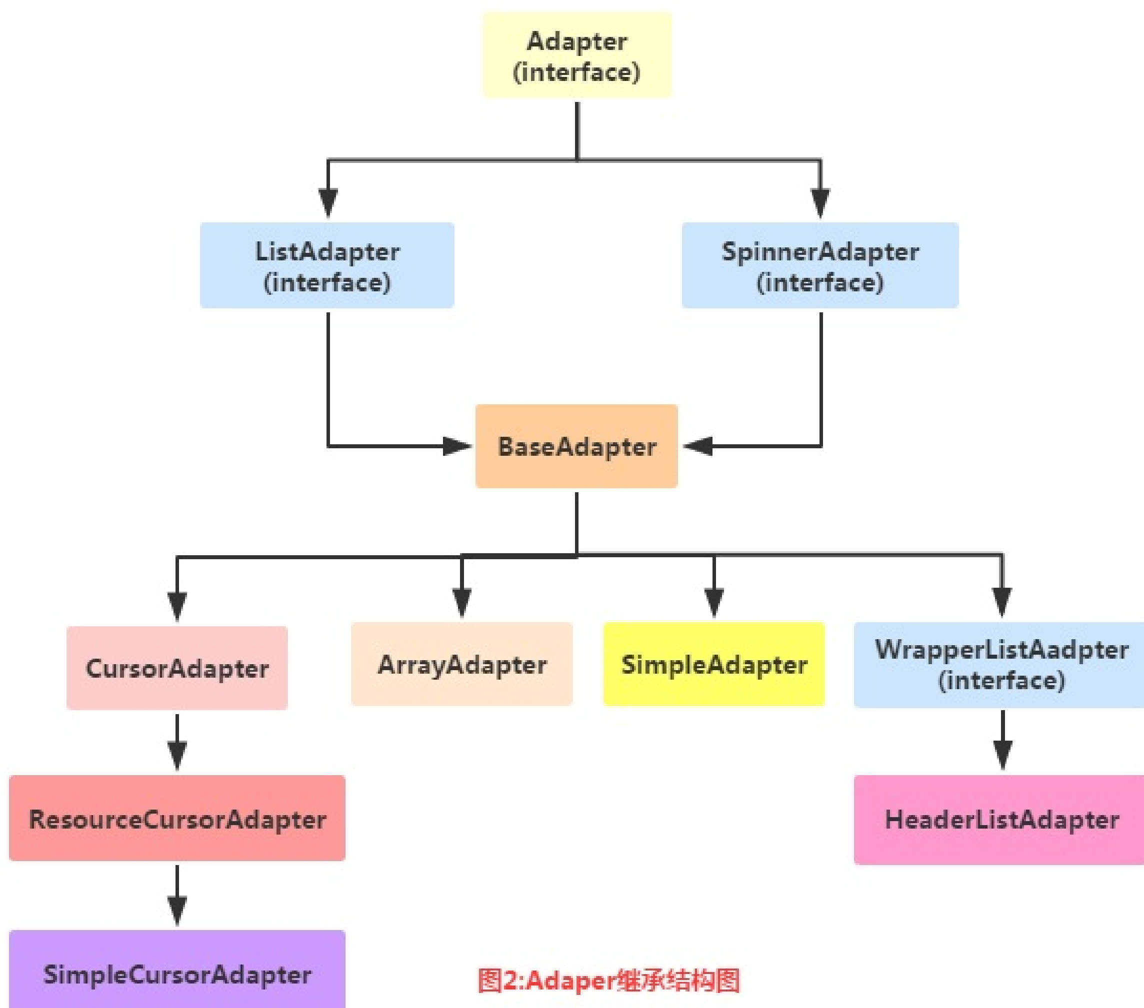


图2:Adaper继承结构图

上面就是Adapter以及继承结构图了，接着我们介绍一下实际开发中还用到的几个Adapter吧！

> - **BaseAdapter**：抽象类，实际开发中我们会继承这个类并且重写相关方法，用得最多的一个Adapter！
> - **ArrayAdapter**：支持泛型操作，最简单的一个Adapter，只能展现一行文字~
> - **SimpleAdapter**：同样具有良好扩展性的一个Adapter，可以自定义多种效果！
> - **SimpleCursorAdapter**：用于显示简单文本类型的listView，一般在数据库那里会用到，不过有点过时， 不推荐使用！

其实一个BaseAdapter就够玩的了，至于其他的，实际开发中用得不多，后面用到在讲解~

3.代码示例：

好的，多说无益，写代码最实际，接下来我们来用写几个简单的Adapter实例， 帮助我们了解Adapter给我们带来的便利， 另外，因为Adapter需要结合ListView, GridView等等控件讲解，一些高级一点的用法我们都放在ListView那里讲！ 这里就简单演示下效果， 另外这里用到的控件是ListView，下一节就会讲解， 现在看不懂也没关系！

1）ArrayAdapter使用示例：

运行效果图：



代码实现：

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //要显示的数据
        String[] strs = {"基神","B神","翔神","曹神","J神"};
        //创建ArrayAdapter
        ArrayAdapter<String> adapter = new ArrayAdapter<String>
                (this,android.R.layout.simple_expandable_list_item_
        //获取ListView对象，通过调用setAdapter方法为ListView设置Adapter
        ListView list_test = (ListView) findViewById(R.id.list_test
        list_test.setAdapter(adapter);
    }
}
```

一些相关的东西：

1.除了通过数组外，我们还可以写到一个数组资源文件中：

比如：在res\valuse下创建一个数组资源的xml文件：**arrays.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="myarray">
    <item>语文</item>
    <item>数学</item>
    <item>英语</item>
    </string-array>
</resources>
```

接着布局的listview属性设置下这个列表项：

```
<ListView
        android:id="@id/list_test"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:entries="@array/myarray"/>
```

就可以了~

当然我们也可以在Java代码中这样写：

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResourc
        R.array.myarray,android.R.layout.simple_list_item_multiple_
```

同样也是可以的！

2.一开始也说了这个ArrayAdapter支持泛型，那么集合必不可少啦，比如，这样写：

```
List<String> data = new ArrayList<String>();
data.add("基神");
data.add("B神");
ArrayAdapter<String> adapter = new ArrayAdapter<String>
                (this,android.R.layout.simple_expandable_list_item_
```

就可以了~

3.我们看到了在实例化ArrayAdapter的第二个参数：**android.R.layout.simple_expandable_list_item_1** 其实这些是系统给我们提供好的一些ListView模板，有下面几种：

> **simple_list_item_1**：单独一行的文本框
>
> 语文
>
> **simple_list_item_2**：两个文本框组成
>
> 慳曳
> 公的
>
> **simple_list_item_checked**：每项都是由一个已选中的列表项
>
> 语文
>
> **simple_list_item_multiple_choice**：都带有一个复选框
>
> 语文
>
> **simple_list_item_single_choice**：都带有一个单选钮
>
> 语文
> /blog.csdn.net/coder_pig

2）SimpleAdapter使用示例：

SimpleAdapter：简单的Adapter，看似简单，功能强大，下面我们来写个稍微复杂一点的列表 布局吧！

运行效果图：



代码实现：

先来编写一个列表项目每一项的布局：

list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <!-- 定义一个用于显示头像的ImageView -->
    <ImageView
        android:id="@+id/imgtou"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:baselineAlignBottom="true"
        android:paddingLeft="8dp" />

    <!-- 定义一个竖直方向的LinearLayout,把QQ呢称与说说的文本框设置出来 -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="8dp"
            android:textColor="#1D1D1C"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/says"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="8px"
            android:textColor="#B4B4B9"
            android:textSize="14sp" />

    </LinearLayout>

</LinearLayout>
```

接下来是**MainActivity.java**:

```
public class MainActivity extends AppCompatActivity {

    private String[] names = new String[]{"B神", "基神", "曹神"};
    private String[] says = new String[]{"无形被黑，最为致命", "大神好
    private int[] imgIds = new int[]{R.mipmap.head_icon1, R.mipmap

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        List<Map<String, Object>> listitem = new ArrayList<Map<Str
        for (int i = 0; i < names.length; i++) {
            Map<String, Object> showitem = new HashMap<String, Obje
            showitem.put("touxiang", imgIds[i]);
            showitem.put("name", names[i]);
            showitem.put("says", says[i]);
            listitem.add(showitem);
        }

        //创建一个simpleAdapter
        SimpleAdapter myAdapter = new SimpleAdapter(getApplication
        ListView listView = (ListView) findViewById(R.id.list_test
        listView.setAdapter(myAdapter);
    }
}
```

好的，上面就是SimpleAdapter的简单用法了，有点意思~

3）SimpleCursorAdapter使用示例：

虽然这东西过时了，不过对于不怎么会SQLite的初学者来说，用起来还是蛮方便的！记得前面我们学ContentProivder写过的读取联系人的例子么？之前是通过打印Log的 方式显示出来，现在我们通过这个SimpleCursorAdapter把它显示到ListView上！

实现效果图：



代码实现：

先写下listView每个item的布局：

list_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/list_name"
        android:layout_width="0dp"
        android:layout_height="64dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="小猪"
        android:textColor="#0000FF"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/list_phone"
        android:layout_width="0dp"
        android:layout_height="64dp"
        android:layout_weight="1"
        android:gravity="center"
        android:text="13798989898"
        android:textColor="#EA5C4D"
        android:textSize="18sp" />

</LinearLayout>
```

接着activity_main布局和前面的一样，就是简单的ListView，然后是

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView list_test = (ListView) findViewById(R.id.list_test
        //读取联系人
        Cursor cursor = getContentResolver()
                .query(ContactsContract.CommonDataKinds.Phone.CONTE
        SimpleCursorAdapter spcAdapter = new SimpleCursorAdapter(th
                new String[]{ContactsContract.CommonDataKinds.Phone
                new int[]{R.id.list_name,R.id.list_phone});
        list_test.setAdapter(spcAdapter);
    }
}
```

最后AndroidManifest.xml里加个读联系人的权限就可以了！

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

一问一答：

> 问：就这么简单？——答：是的，直接获取到Cursor，然后绑定就好了，无需你自己再写什么SQL语句！ 问：你说这东西过时了，那拿什么来代替？——答：一般的做法是自己重写BaseAdapter，获取到数据集合后跟对应的控件进行绑定！ 问：这个SimpleCursorAdapter还有没有要注意的地方？——答：有，使用SimpleCursorAdapter的话,绑定的数据库表中一定要有*id*这个字段, 或者*as* id;而且在绑定时取出的数据必须包含这个*id*项,否则的话会报以下错误! **java.lang.IllegalArgumentException: column '**id' does not exist**

本节小结：

> 好的，关于Adapter的基础讲解就到这里，当然我们这里讲解的三个Adapter，我们实际开发中... 基本上用不到，哈哈，除了SimpleAdapter偶尔可能会用下，一般我们都是重写BaseAdapter的！ 另外，关于BaseAdapter的，有很多东西要讲解，就把他丢到ListView那里一起讲，毕竟Adapter 总是和View沾边，而且ListView是我们用得最多的一个空间~嗯，本节就到这里，谢谢~

2.4.5 ListView简单实用

本节引言：

> 本节我们来继续学习没有讲完的UI控件部分， 回顾上一节，我们介绍了Adapter适配器的概念，然后学习了三个最简单的适配器的使用：ArrayAdapter，SimpleAdapter和SimpleCursorAdapter，而本节给大家讲解的是第一个 需搭配Adapter使用的UI控件：ListView，不过在版本中被RecyclerView这个新控件替换掉了！ 列表作为最常用的控件之一，还是有必要好好学习的，本节以一个初学者的角度来学习 ListView，ListView的属性，以及BaseAdapter简单定义，至于ListView优化这些， 我们一步步来~莫急！

1.自定义BaseAdapter，然后绑定ListView的最简单例子

先看看我们要实现的效果图：



一个很简单的ListView，自己写下Item，然后加载点数据这样~ 下面贴下关键代码：

Animal.java:

```
/**
 * Created by Jay on 2015/9/18 0018.
 */  public  class  Animal  {  private  String aName;  private  Str
```

AnimalAdapter.java：自定义的BaseAdapter：

```
/**
 * Created by Jay on 2015/9/18 0018.
 */  public  class  AnimalAdapter  extends  BaseAdapter  {  private
```

最后是**MainActivity.java**：

```
public  class  MainActivity  extends  AppCompatActivity  {  private
```

好的，自定义BaseAdapter以及完成数据绑定就是这么简单~ 别问我拿示例的代码，刚开始学就会写出这些代码，我只是演示下流程，让大家熟悉 熟悉而已~另外，也是为下面的属性验证做准备~

2.表头表尾分割线的设置：

> listview作为一个列表控件，他和普通的列表一样，可以自己设置表头与表尾：以及分割线，可供我们设置的属性如下：
>
> - **footerDividersEnabled**：是否在footerView(表尾)前绘制一个分隔条,默认为true
> - **headerDividersEnabled**:是否在headerView(表尾)前绘制一个分隔条,默认为true
> - **divider**:设置分隔条,可以用颜色分割,也可以用drawable资源分割
> - **dividerHeight**:设置分隔条的高度
>
> 翻遍了了API发现并没有可以直接设置ListView表头或者表尾的属性，只能在Java中写代码 进行设置了，可供我们调用的方法如下：
>
> - **addHeaderView(View v)**：添加headView(表头),括号中的参数是一个View对象
> - **addFooterView(View v)**：添加footerView(表尾)，括号中的参数是一个View对象
> - **addHeaderView(headView, null, false)**：和前面的区别：设置Header是否可以被选中
> - **addFooterView(View,view,false)**：同上
>
> 对了，使用这个addHeaderView方法必须放在listview.setAdapter前面，否则会报错。

使用示例：

运行效果图：



代码实现：

先编写下表头与表尾的布局：

view_header.xml(表头),表尾一样，就不贴了：

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:andro
```

MainActivty.java:

```
public class MainActivity extends AppCompatActivity implements
```

好的，代码还是比较简单的，从上面我们看出来一个要注意的问题，就是：

> 添加表头表尾后，我们发现positon是从表头开始算的，就是你添加的第一个数据本来的postion是 0，但是此时却变成了0，因为表头也算！！

3.列表从底部开始显示：stackFromBottom

> 如果你想让列表显示你列表的最下面的话，那么你可以使用这个属性，将stackFromBottom 属性设置为true即可，设置后的效果图如下：



4.设置点击颜色cacheColorHint

> 如果你为ListView设置了一个图片作为Background的话，当你拖动或者点击listView空白位置会发现 item都变成黑色了，这是时候我们可以通过这个**cacheColorHint**将颜色设置为透明:#00000000

5.隐藏滑动条

> 我们可以通过设置：android:scrollbars="none" 或者setVerticalScrollBarEnabled(true); 解决这个问题！

本节小结：

> 好的，关于ListView的基本用法大概就这些，当然除了上述的这些属性外还有其他的， 实际遇到再查查吧~这里知道如何去重写BaseAdapter和完成数据绑定就好，下节我们来 教大家如何来优化这个BaseAdapter的编写~

2.4.6 BaseAdapter优化

本节引言：

> 上一节中我们学习了如何来使用一个ListView以及自定义一个简单的BaseAdapter，我们从代码中可以看出比较重要的两个方法:getCount()和getView()，界面上有多少列就会调用多少次getView，这个时候可能看出一些端倪，每次都是新inflate一个View，都要进行这个XML的解析，这样会很浪费资源，当然，几十列或者几百列的列表并不能体现什么问题，但假如更多或者布局更加复杂？所以学习ListView的优化很重要，而本节针对的是BaseAdapter的优化，优化的两点有，复用convertView 以及使用ViewHolder重用组件，不用每次都findViewById，我们具体通过代码来体会吧！

1.复用ConvertView：

> 上面也说了，界面上有多少个Item，那么getView方法就会被调用多少次！我们来看看上一节我们写的getView()部分的代码：

```
@Override
public View getView(int position, View convertView, ViewGroup paren
    convertView = LayoutInflater.from(mContext).inflate(R.layout.it
    ImageView img_icon = (ImageView) convertView.findViewById(R.id.
    TextView txt_aName = (TextView) convertView.findViewById(R.id.t
    TextView txt_aSpeak = (TextView) convertView.findViewById(R.id.

    img_icon.setBackgroundResource(mData.get(position).getaIcon());
    txt_aName.setText(mData.get(position).getaName());
    txt_aSpeak.setText(mData.get(position).getaSpeak());
    return convertView;
}
```

> 是吧，inflate()每次都要加载一次xml，其实这个convertView是系统提供给我们的可供服用的View 的缓存对象，那就坐下判断咯，修改下，优化后的代码：

```
@Override
public View getView(int position, View convertView, ViewGroup paren

    if(convertView == null){
        convertView = LayoutInflater.from(mContext).inflate(R.layou
    }

    ImageView img_icon = (ImageView) convertView.findViewById(R.id.
    TextView txt_aName = (TextView) convertView.findViewById(R.id.t
    TextView txt_aSpeak = (TextView) convertView.findViewById(R.id.

    img_icon.setBackgroundResource(mData.get(position).getaIcon());
    txt_aName.setText(mData.get(position).getaName());
    txt_aSpeak.setText(mData.get(position).getaSpeak());
    return convertView;
}
```

2.ViewHolder重用组件

> 嘿嘿，getView()会被调用多次，那么findViewById不一样得调用多次，而我们的ListView的Item 一般都是一样的布局，我们可以对这里在优化下，我们可以自己定义一个ViewHolder类来对这一部分 进行性能优化！修改后的代码如下：

```
@Override
public View getView(int position, View convertView, ViewGroup paren
    ViewHolder holder = null;
    if(convertView == null){
        convertView = LayoutInflater.from(mContext).inflate(R.layou
        holder = new ViewHolder();
        holder.img_icon = (ImageView) convertView.findViewById(R.ic
        holder.txt_aName = (TextView) convertView.findViewById(R.ic
        holder.txt_aSpeak = (TextView) convertView.findViewById(R.i
        convertView.setTag(holder);   //将Holder存储到convertView中
    }else{
        holder = (ViewHolder) convertView.getTag();
    }
    holder.img_icon.setBackgroundResource(mData.get(position).getal
    holder.txt_aName.setText(mData.get(position).getaName());
    holder.txt_aSpeak.setText(mData.get(position).getaSpeak());
    return convertView;
}

static class ViewHolder{
    ImageView img_icon;
    TextView txt_aName;
    TextView txt_aSpeak;
}
```

> 没错就是这么简单，你以后BaseAdapter照着这个模板写就对了，哈哈，另外这个修饰ViewHolder的 static，关于是否定义成静态，跟里面的对象数目是没有关系的，加静态是为了在多个地方使用这个 Holder的时候，类只需加载一次，如果只是使用了一次，加不加也没所谓！——**Berial(B神)**原话~

本节小结：

> 好的，关于BaseAdapter的优化大概就上述的两种，非常简单，复用ConvertView以及自定义ViewHolder 减少findViewById()的调用~如果你有其他关于BaseAdapter优化的建议欢迎提出，谢谢~

2.4.7ListView的焦点问题

本节引言

> 如果你往ListView的Item中添加了Button，CheckBox，EditText等控件的话，你可能需要考虑 到一个问题：ListView的一个焦点问题！本节我们就来学习下解决这个问题的几个方法！
>
> 我们可以写个简答的listView，上面有一个Button，CheckBox，EditText，但是当我们点击发现， ListView的item点击不了，触发不了onItemClick的方法，也触发不了onItemLongClick方法， 这个就是ListView的一个焦点问题了！就是ListView的焦点被其他控件抢了，下面我们来看看如何 解决这个问题？

方法1：为抢占了控件的组件设置:android:focusable="false"

> 如题，只需为抢占了ListView Item焦点的控件设置**android:focusable="false"**即可解决这个问题 或者在代码中获得控件后调用：**setFocusable(false)** !!另外，EditText却不行，如果我们设置了android:focusable="false"，这B可以获取焦点但是一下子 又失去了焦点，而且也不会弹出小键盘，暂不知道如何解决，听别人说是ListView的一个bug，如果有知道解决方法的欢迎告知下，谢谢~

方法2：item根节点设置android:descendantFocusability="blocksDescendants"

> 如题，在Item布局的根节点添加上述属性，**android:descendantFocusability="blocksDescendants"** 即可，另外该属性有三个可供选择的值：
>
> - **beforeDescendants**：viewgroup会优先其子类控件而获取到焦点
> - **afterDescendants**：viewgroup只有当其子类控件不需要获取焦点时才获取焦点
> - **blocksDescendants**：viewgroup会覆盖子类控件而直接获得焦点

本节小结：

> 好的，以上就是解决ListView焦点问题的两个方法，非常简单，如果有关于EditText 焦点问题解决方案的欢迎提出，谢谢~

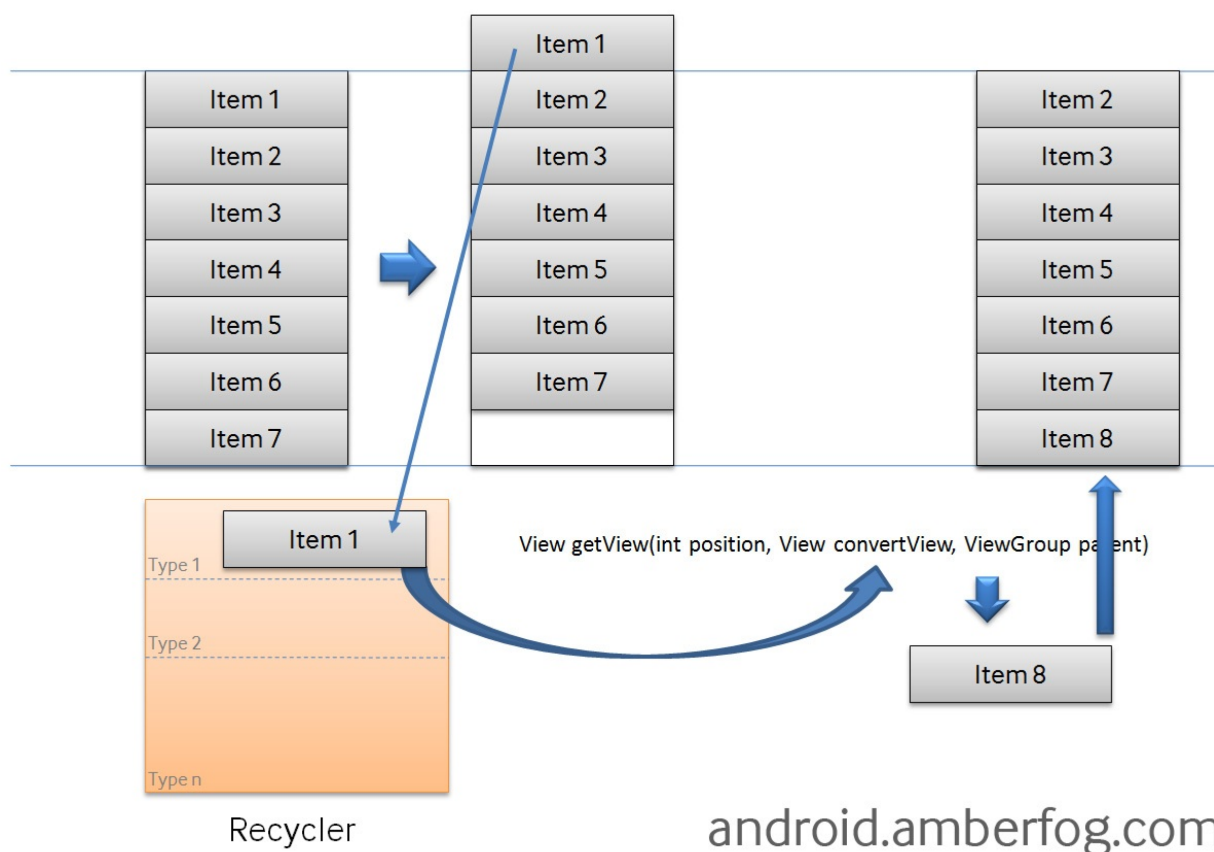
2.4.8 ListView之checkbox错位问题解决

本节引言：

> 作为ListView经典问题之一，如果你尝试过自定义ListView的item，在上面带有一个checkbox的话，那么 当你的item数超过了一页的话，就会出现这个问题，下面我们来分析下出现这种问题的原因，以及如何来 解决这个问题！

1.问题发生的原因：

这是网上找来的一幅关于ListView getView方法调用机制的一个图



上图中有一个**Recycler**的东东，平时我们ListView上可见的Item处于内存中，而且他的Item则放在 这个Recycler中，第一次加载item时，当前页面中的convertView都为NULL，当滚出屏幕，这是时候 **ConvertView**不为空，所以新的一项会复用这个ConvertView！我们可以写个简单的例子，跟下log，下面是运行后的一些Log图！

```
09-21 14:32:21.781  23093-23093/? E/HEHE :  getView() + 0 + null
09-21 14:32:21.783  23093-23093/? E/HEHE :  getView() + 1 + null
09-21 14:32:21.786  23093-23093/? E/HEHE :  getView() + 2 + null
09-21 14:32:21.788  23093-23093/? E/HEHE :  getView() + 3 + null
09-21 14:32:21.790  23093-23093/? E/HEHE :  getView() + 4 + null
09-21 14:32:21.792  23093-23093/? E/HEHE :  getView() + 5 + null
09-21 14:32:21.795  23093-23093/? E/HEHE :  getView() + 6 + null
09-21 14:32:21.797  23093-23093/? E/HEHE :  getView() + 7 + null
09-21 14:32:21.799  23093-23093/? E/HEHE :  getView() + 8 + null
09-21 14:32:21.802  23093-23093/? E/HEHE :  getView() + 9 + null
09-21 14:32:21.804  23093-23093/? E/HEHE :  getView() + 10 + null
09-21 14:32:27.606  23093-23093/? E/HEHE :  getView() + 11 + null
09-21 14:32:30.134  23093-23093/? E/HEHE :  getView() + 12 + android.widget.LinearLayout{278192cc V.E..... ........ 0,-144-1080,0}
09-21 14:32:31.038  23093-23093/? E/HEHE :  getView() + 13 + android.widget.LinearLayout{22add915 V.E..... ........ 0,-143-1080,1}
09-21 14:32:37.947  23093-23093/? E/HEHE :  getView() + 14 + android.widget.LinearLayout{3768042a V.E..... ........ 0,-139-1080,5}
```

从图中看出，Postion从12开始，ConvertView就不为空了，具体这里代表的是什么， 我也不知道，目测要走源码...我们知道这里ConvertView会缓存就好，就是因为这个原因 造成的checkbox错位，所以第一个解决方式就是，不重用这个ConvertView，或者 说每次getView都将这个ConvertView设置为null，但是如果需要显示的Item数目巨大的话， 这种方法就会显得非常臃肿，一般实际开发我们使用的是下面的解决方法： 找个东 东来保存当前**Item CheckBox**的状态，初始化的时候进行判断，设置是否选中！

2.解决方法示例：

> 好的存储这个Checkbox的方法有很多，你可以放到一个HashMap<Integer, Boolean>中， 每次初始化的时候根据postion取出对应的boolean值，然后再进行checkbox的状态设置； 而笔者的做法则是在entity类中加入了一个boolean值用于判断，下面是笔者一个项目中 抽取出来的代码，代码比较简单，相信你看完会秒懂的~

Entity类：**Person.java**：

```
public class Person implements Serializable{
    private String name;
    private String number;
    private boolean checkStatus;

    public Person(String name, String number) {
        super();
        this.name = name;
        this.number = number;
        this.checkStatus = false;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    public boolean getCheckStatus() {
        return checkStatus;
    }

    public void setCheckStatus(boolean checkStatus) {
        this.checkStatus = checkStatus;
    }

}
```

实现的Adapter类：**ContactListAdapter.java**：**

```
public class ContactListAdapter extends BaseAdapter implements Comp

    private List<Person> mData;
    private Context mContext;

    public ContactListAdapter(List<Person> data, Context context) {
        mData = data;
        mContext = context;
```

```
    }

    // 定义一个刷新数据的方法
    public void changeData(List<Person> data) {
        mData = data;
        notifyDataSetChanged();
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Person getItem(int position) {
        return mData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        final int index = position;
        ViewHolder viewHolder;
        if (convertView == null) {
            convertView = LayoutInflater.from(mContext).inflate(
                    R.layout.item_contact, parent, false);
            viewHolder = new ViewHolder();
            viewHolder.ly = (RelativeLayout) convertView
                    .findViewById(R.id.lyContactListItem);
            viewHolder.txtName = (TextView) convertView
                    .findViewById(R.id.txtName);
            viewHolder.txtNumber = (TextView) convertView
                    .findViewById(R.id.txtNumber);
            viewHolder.cbxStatus = (CheckBox) convertView
                    .findViewById(R.id.cbxStatus);
            convertView.setTag(viewHolder);
            viewHolder.cbxStatus.setTag(index);
        } else {
            viewHolder = (ViewHolder) convertView.getTag();
        }
        viewHolder.cbxStatus.setOnCheckedChangeListener(this);
        viewHolder.cbxStatus.setChecked(mData.get(position).getchec
        viewHolder.txtName.setText(mData.get(index).getName());
        viewHolder.txtNumber.setText(mData.get(index).getNumber());
        return convertView;
    }

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean
```

```
            int index = (int)buttonView.getTag();
            if (isChecked)
                mData.get(index).setCheckStatus(true);
            else
                mData.get(index).setCheckStatus(false);
        }

        private class ViewHolder {
            RelativeLayout ly;
            TextView txtName;
            TextView txtNumber;
            CheckBox cbxStatus;
        }
    }
```

嘿嘿，非常简单，另外别忘了一点： **checkbox**监听器的方法要添加在初始化**Checkbox**状态的代码之前哦~

本节引言：

> 好的，本节给大家讲解了ListView的一个经典问题，ListView中checkbox错位的 问题解决，只需简单的添加一个记录checkbox选择状态的值，然后重写checkbox 点击事件的时候，先做判断~谢谢~

2.4.9 ListView的数据更新问题

本节引言：

> 我们前面已经学习了ListView的一些基本用法咧，但是细心的你可能发现了，我们的数据 一开始定义好的，都是静态的，但是实际开发中，我们的数据往往都是动态变化的，比如 我增删该了某一列，那么列表显示的数据也应该进行同步的更新，那么本节我们就来探讨 下ListView数据更新的问题，包括全部更
>
> 新，以及更新其中的一项，那么开始本节内容！~ 

1.先写个正常的demo先

> 好的，先写个正常的Demo先，等下我们再慢慢调：

entity类：**Data.java**：

```
/**
 * Created by Jay on 2015/9/21 0021.
 */
public class Data {
    private int imgId;
    private String content;

    public Data() {}

    public Data(int imgId, String content) {
        this.imgId = imgId;
        this.content = content;
    }

    public int getImgId() {
        return imgId;
    }

    public String getContent() {
        return content;
    }

    public void setImgId(int imgId) {
        this.imgId = imgId;
    }

    public void setContent(String content) {
        this.content = content;
    }
}
```

Activity布局以及列表项布局：

activity_main.xml：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/list_one"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

item_list.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/img_icon"
        android:layout_width="56dp"
        android:layout_height="56dp"/>

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="10dp"
        android:textSize="18sp" />

</LinearLayout>
```

自定义BaseAdapter的实现：**MyAdapter.java**：

```
/**
 * Created by Jay on 2015/9/21 0021.
 */
public class MyAdapter extends BaseAdapter {

    private Context mContext;
    private LinkedList<Data> mData;

    public MyAdapter() {}

    public MyAdapter(LinkedList<Data> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }
```

```
    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        ViewHolder holder = null;
        if(convertView == null){
            convertView = LayoutInflater.from(mContext).inflate(R.l
            holder = new ViewHolder();
            holder.img_icon = (ImageView) convertView.findViewById(
            holder.txt_content = (TextView) convertView.findViewByI
            convertView.setTag(holder);
        }else{
            holder = (ViewHolder) convertView.getTag();
        }
        holder.img_icon.setImageResource(mData.get(position).getImg
        holder.txt_content.setText(mData.get(position).getContent()
        return convertView;
    }

    private class ViewHolder{
        ImageView img_icon;
        TextView txt_content;
    }

}
```

MainActivity.java的编写：

```
public class MainActivity extends AppCompatActivity {

    private ListView list_one;
    private MyAdapter mAdapter = null;
    private List<Data> mData = null;
    private Context mContext = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        bindViews();
        mData = new LinkedList<Data>();
        mAdapter = new MyAdapter((LinkedList<Data>) mData,mContext)
        list_one.setAdapter(mAdapter);
    }

    private void bindViews(){
        list_one = (ListView) findViewById(R.id.list_one);
    }

}
```

可以运行，运行后发现我们的页面并没有任何的数据，白茫茫的一片，这样的用户体验并不好， 我们可以通过调用ListView的一个**setEmptyView(View)**的方法，当ListView数据为空的时候， 显示一个对应的View，另外发现这个方法很奇葩，动态添加的View，竟然无效，只能在ListView 所在的布局文件中添加当ListView无数据时，想显示的View，另外用这个setEmptyView设置后的 View，加载的时候竟然不会显示出来，好灵异....比如这里的是没有数据时显示一个没有数据 的TextView，部分代码如下：

```
<TextView
        android:id="@+id/txt_empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:textSize="15pt"
        android:textColor="#000000"/>

txt_empty = (TextView) findViewById(R.id.txt_empty);
txt_empty.setText("暂无数据~");
list_one.setEmptyView(txt_empty);
```

当然除了这种方法外我们还可以定义一个与ListView一样大小位置的布局，然后设置， android:visibility="gone"，在Java代码中对mData集合的size进行判断，如果==0， 说明没数据，让这个布局显示出来，当有数据的时候让这个布局隐藏~

2.添加一条记录

好的，我们弄个添加按钮，没按一次添加一条记录哈~

运行效果图:



代码实现

在我们自定义的BaseAdapter中定义一个方法，方法内容如下：

```
public void add(Data data) {
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(data);
    notifyDataSetChanged();
}
```

然后布局自己加个按钮，然后设置下事件，代码如下：

```
private Button btn_add;
btn_add = (Button) findViewById(R.id.btn_add);
btn_add.setOnClickListener(this);

@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.btn_add:
            mAdapter.add(new Data(R.mipmap.ic_icon_qitao,"给猪哥跪了
            flag++;
            break;
    }
}
```

嘿嘿，成了，添加数据就这么简单~，如果你想插入到特定位置，也行，我们Adapter类里，再另外 写一个方法：

```
//往特定位置，添加一个元素
public void add(int position,Data data){
    if (mData == null) {
        mData = new LinkedList<>();
    }
    mData.add(position,data);
    notifyDataSetChanged();
}
```

然后加个按钮，写个事件：

```
private Button btn_add2;
btn_add2 = (Button) findViewById(R.id.btn_add2);
btn_add2.setOnClickListener(this);

case R.id.btn_add2:
//position从0开始算的
mAdapter.add(4,new Data(R.mipmap.ic_icon_qitao,"给猪哥跪了~~~ x " +
break;
```

运行效果图：



可以看到我们的第九项插入到了第五个位置~

3.删除某一项

同样的，我们写两个方法，一个直接删对象，一个根据游标来删：

```
public void remove(Data data) {
    if(mData != null) {
        mData.remove(data);
    }
    notifyDataSetChanged();
}

public void remove(int position) {
    if(mData != null) {
        mData.remove(position);
    }
    notifyDataSetChanged();
}
```

然后加两个Button，调用下这两个方法：

```
case R.id.btn_remove:
    mAdapter.remove(mData_5);
    break;
case R.id.btn_remove2:
    mAdapter.remove(2);
    break;
```

运行效果图：



从图中我们可以看到，第五项被移除了，然后点击游标删除数据，一直删的是第三项！

4.移除所有的记录：

> 这个更加简单，直接调用clear方法即可！方法代码如下：

```
public void clear() {
    if(mData != null) {
        mData.clear();
    }
    notifyDataSetChanged();
}
```

5.更新某一个记录

细心的你应该发现了，进行了数据修改操作后，都会调用一个notifyDataSetChanged(); 一开始我以为：

> notifyDataSetChanged()会把界面上现实的的item都重绘一次，这样会影响ui性能吧，如果数据量 很大，但是我改变一项就要重新绘制所有的item，这肯定不合理是吧！于是乎，我用了一个傻办法 来修改某个Item中控件的值，我在Java代码中写了这样一段代码：

```
private void updateListItem(int postion,Data mData){
    int visiblePosition = list_one.getFirstVisiblePosition();
    View v = list_one.getChildAt(postion - visiblePosition);
    ImageView img = (ImageView) v.findViewById(R.id.img_icon);
    TextView tv = (TextView) v.findViewById(R.id.txt_content);
    img.setImageResource(mData.getImgId());
    tv.setText(mData.getContent());
}
```

后来和群里的朋友讨论了下，发现自己错了：

> notifyDataSetChanged()方法会判断是否需要重新渲染，如果当前item没有必要重新渲染 是不会重新渲染的，如果某个Item的状态发生改变，都会导致View的重绘，而重绘的并不是 所有的Item，而是View状态发生变化的那个Item！所以我们直接notifyDataSetChange()方法 即可，当然知道多一个上面的方法也没什么~

代码下载：

ListViewDemo3.zip

本节小结：

> 好的，本节跟大家讲述了ListView中数据更新的实现，当然不止ListView，其他的Adapter 类控件都可以调用这些方法来完成数据更新~就说这么多吧~谢谢

2.5.0 构建一个可复用的自定义BaseAdapter

本节引言：

> 如题，本节给大家带来的是构建一个可复用的自定义BaseAdapter，我们每每涉及到ListView GridView等其他的Adapter控件，都需要自己另外写一个BaseAdapter类，这样显得非常麻烦， 又比如，我们想在一个界面显示两个ListView的话，我们也是需要些两个BaseAdapter... 这，程序员都是喜欢偷懒的哈，这节我们就来写一个可复用的自定义BaseAdapter类~

1.我们一点点开始改：

首先我们把上节写的自定义BaseAdapter贴下，等下我们就要对他进行升级改造

```
/**
 * Created by Jay on 2015/9/21 0021.
 */
public class MyAdapter extends BaseAdapter {

    private Context mContext;
    private LinkedList<Data> mData;

    public MyAdapter() {
    }

    public MyAdapter(LinkedList<Data> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
```

```
    public View getView(int position, View convertView, ViewGroup p
        ViewHolder holder = null;
        if (convertView == null) {
            convertView = LayoutInflater.from(mContext).inflate(R.l
            holder = new ViewHolder();
            holder.img_icon = (ImageView) convertView.findViewById(
            holder.txt_content = (TextView) convertView.findViewByI
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        holder.img_icon.setImageResource(mData.get(position).getImg
        holder.txt_content.setText(mData.get(position).getContent()
        return convertView;
    }

    //添加一个元素
    public void add(Data data) {
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(data);
        notifyDataSetChanged();
    }

    //往特定位置，添加一个元素
    public void add(int position,Data data){
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(position, data);
        notifyDataSetChanged();
    }

    public void remove(Data data) {
        if(mData != null) {
            mData.remove(data);
        }
        notifyDataSetChanged();
    }

    public void remove(int position) {
        if(mData != null) {
            mData.remove(position);
        }
        notifyDataSetChanged();
    }

    public void clear() {
        if(mData != null) {
            mData.clear();
        }
        notifyDataSetChanged();
```

```
    }

    private class ViewHolder {
        ImageView img_icon;
        TextView txt_content;
    }

}
```

升级1：将Entity设置成泛型

好的，毕竟我们传递过来的Entitiy实体类可能千奇百怪，比如有Person，Book，Wether等，所以我们 将Entity设置成泛型，修改后的代码如下：

```
<pre>
public class MyAdapter<T> extends BaseAdapter {

    private Context mContext;
    private LinkedList<T> mData;

    public MyAdapter() {
    }

    public MyAdapter(LinkedList<T> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        ViewHolder holder = null;
        if (convertView == null) {
            convertView = LayoutInflater.from(mContext).inflate(R.l
            holder = new ViewHolder();
```

```
            holder.img_icon = (ImageView) convertView.findViewById(
            holder.txt_content = (TextView) convertView.findViewByI
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        holder.img_icon.setImageResource(mData.get(position).getImg
        holder.txt_content.setText(mData.get(position).getContent()
        return convertView;
    }

    //添加一个元素
    public void add(T data) {
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(data);
        notifyDataSetChanged();
    }

    //往特定位置，添加一个元素
    public void add(int position,T data){
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(position, data);
        notifyDataSetChanged();
    }

    public void remove(T data) {
        if(mData != null) {
            mData.remove(data);
        }
        notifyDataSetChanged();
    }

    public void remove(int position) {
        if(mData != null) {
            mData.remove(position);
        }
        notifyDataSetChanged();
    }

    public void clear() {
        if(mData != null) {
            mData.clear();
        }
        notifyDataSetChanged();
    }

    private class ViewHolder {
        ImageView img_icon;
        TextView txt_content;
```

```
    }

}
```

好的，上面我们做的事仅仅是将Data类型换成了泛型T！

升级2：ViewHolder类的升级改造：

> 我们先来看看前面我们的ViewHolder干了什么？ 答：findViewById，设置控件状态； 下面我们想在完成这个基础上，将getView()方法大部分的逻辑写到ViewHolder类里， 这个ViewHolder要做的事：
>
> - 定义一个查找控件的方法，我们的思路是通过暴露公共的方法，调用方法时传递过来 控件id，以及设置的内容，比如TextView设置文本： public ViewHolder setText(int id, CharSequence text){文本设置}
> - 将convertView复用部分搬到这里，那就需要传递一个context对象了，我们把需要获取 的部分都写到构造方法中！
> - 写一堆设置方法(public)，比如设置文字大小颜色，图片背景等！

好的，接下来我们就来一步步改造我们的ViewHolder类

1）相关参数与构造方法：

```
public static class ViewHolder {

    private SparseArray<View> mViews;   //存储ListView 的 item中的Vi
    private View item;                  //存放convertView
    private int position;               //游标
    private Context context;            //Context上下文

    //构造方法，完成相关初始化
    private ViewHolder(Context context, ViewGroup parent, int layou
        mViews = new SparseArray<>();
        this.context = context;
        View convertView = LayoutInflater.from(context).inflate(lay
        convertView.setTag(this);
        item = convertView;
    }

    ImageView img_icon;
    TextView txt_content;
}
```

2）绑定ViewHolder与Item

在上面的基础上我们再添加一个绑定的方法

```
//绑定ViewHolder与item
public static ViewHolder bind(Context context, View convertView, Vi
                              int layoutRes, int position) {
    ViewHolder holder;
    if(convertView == null) {
        holder = new ViewHolder(context, parent, layoutRes);
    } else {
        holder = (ViewHolder) convertView.getTag();
        holder.item = convertView;
    }
    holder.position = position;
    return holder;
}
```

3）根据id获取集合中保存的控件

```
public <T extends View> T getView(int id) {
    T t = (T) mViews.get(id);
    if(t == null) {
        t = (T) item.findViewById(id);
        mViews.put(id, t);
    }
    return t;
}
```

4) 接着我们再定义一堆暴露出来的方法

```
/**
 * 获取当前条目
 */
public View getItemView() {
    return item;
}

/**
 * 获取条目位置
 */
public int getItemPosition() {
    return position;
}

/**
 * 设置文字
 */
public ViewHolder setText(int id, CharSequence text) {
```

```
    View view = getView(id);
    if(view instanceof TextView) {
        ((TextView) view).setText(text);
    }
    return this;
}

/**
 * 设置图片
 */
public ViewHolder setImageResource(int id, int drawableRes) {
    View view = getView(id);
    if(view instanceof ImageView) {
        ((ImageView) view).setImageResource(drawableRes);
    } else {
        view.setBackgroundResource(drawableRes);
    }
    return this;
}

/**
 * 设置点击监听
 */
public ViewHolder setOnClickListener(int id, View.OnClickListener l
    getView(id).setOnClickListener(listener);
    return this;
}

/**
 * 设置可见
 */
public ViewHolder setVisibility(int id, int visible) {
    getView(id).setVisibility(visible);
    return this;
}

/**
 * 设置标签
 */
public ViewHolder setTag(int id, Object obj) {
    getView(id).setTag(obj);
    return this;
}

//其他方法可自行扩展
```

好的，ViewHolder的改造升级完成~

升级3：定义一个抽象方法，完成ViewHolder与Data数据集的绑定

```
public abstract void bindView(ViewHolder holder, T obj);
```

我们创建新的BaseAdapter的时候，实现这个方法就好，另外，别忘了把我们自定义 的BaseAdapter改成abstact抽象的！

升级4：修改getView()部分的内容

```
@Override
public View getView(int position, View convertView, ViewGroup parer
    ViewHolder holder = ViewHolder.bind(parent.getContext(), conver
            , position);
    bindView(holder,getItem(position));
    return holder.getItemView();
}
```

2.升级完毕，我们写代码来体验下：

我们要实现的效果图：



就是上面有两个列表，布局不一样，但是我只使用一个BaseAdapter类来完成上述效果！

关键代码如下：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private Context mContext;
    private ListView list_book;
    private ListView list_app;

    private MyAdapter<App> myAdapter1 = null;
    private MyAdapter<Book> myAdapter2 = null;
    private List<App> mData1 = null;
    private List<Book> mData2 = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        init();
```

```
    }

    private void init() {

        list_book = (ListView) findViewById(R.id.list_book);
        list_app = (ListView) findViewById(R.id.list_app);

        //数据初始化
        mData1 = new ArrayList<App>();
        mData1.add(new App(R.mipmap.iv_icon_baidu,"百度"));
        mData1.add(new App(R.mipmap.iv_icon_douban,"豆瓣"));
        mData1.add(new App(R.mipmap.iv_icon_zhifubao,"支付宝"));

        mData2 = new ArrayList<Book>();
        mData2.add(new Book("《第一行代码Android》","郭霖"));
        mData2.add(new Book("《Android群英传》","徐宜生"));
        mData2.add(new Book("《Android开发艺术探索》","任玉刚"));

        //Adapter初始化
        myAdapter1 = new MyAdapter<App>((ArrayList)mData1,R.layout.
            @Override
            public void bindView(ViewHolder holder, App obj) {
                holder.setImageResource(R.id.img_icon,obj.getaIcon(
                holder.setText(R.id.txt_aname,obj.getaName());
            }
        };
        myAdapter2 = new MyAdapter<Book>((ArrayList)mData2,R.layout
            @Override
            public void bindView(ViewHolder holder, Book obj) {
                holder.setText(R.id.txt_bname,obj.getbName());
                holder.setText(R.id.txt_bauthor,obj.getbAuthor());
            }
        };

        //ListView设置下Adapter：
        list_book.setAdapter(myAdapter2);
        list_app.setAdapter(myAdapter1);

    }

}
```

我们写的可复用的BaseAdapter的使用就如上面所述~

3.代码示例下载：

ListViewDemo4.zip

贴下最后写好的MyAdapter类吧，可根据自己的需求进行扩展：

MyAdapter.java：

```
/**
 * Created by Jay on 2015/9/22 0022.
 */
public abstract class MyAdapter<T> extends BaseAdapter {

    private ArrayList<T> mData;
    private int mLayoutRes;           //布局id

    public MyAdapter() {
    }

    public MyAdapter(ArrayList<T> mData, int mLayoutRes) {
        this.mData = mData;
        this.mLayoutRes = mLayoutRes;
    }

    @Override
    public int getCount() {
        return mData != null ? mData.size() : 0;
    }

    @Override
    public T getItem(int position) {
        return mData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        ViewHolder holder = ViewHolder.bind(parent.getContext(), co
                , position);
        bindView(holder, getItem(position));
        return holder.getItemView();
    }

    public abstract void bindView(ViewHolder holder, T obj);

    //添加一个元素
    public void add(T data) {
        if (mData == null) {
            mData = new ArrayList<>();
        }
        mData.add(data);
        notifyDataSetChanged();
    }
```

```
    //往特定位置，添加一个元素
    public void add(int position, T data) {
        if (mData == null) {
            mData = new ArrayList<>();
        }
        mData.add(position, data);
        notifyDataSetChanged();
    }

    public void remove(T data) {
        if (mData != null) {
            mData.remove(data);
        }
        notifyDataSetChanged();
    }

    public void remove(int position) {
        if (mData != null) {
            mData.remove(position);
        }
        notifyDataSetChanged();
    }

    public void clear() {
        if (mData != null) {
            mData.clear();
        }
        notifyDataSetChanged();
    }

    public static class ViewHolder {

        private SparseArray<View> mViews;   //存储ListView 的 item中
        private View item;                  //存放convertView
        private int position;               //游标
        private Context context;            //Context上下文

        //构造方法，完成相关初始化
        private ViewHolder(Context context, ViewGroup parent, int l
            mViews = new SparseArray<>();
            this.context = context;
            View convertView = LayoutInflater.from(context).inflate
            convertView.setTag(this);
            item = convertView;
        }

        //绑定ViewHolder与item
        public static ViewHolder bind(Context context, View convert
                                      int layoutRes, int position)
            ViewHolder holder;
            if (convertView == null) {
                holder = new ViewHolder(context, parent, layoutRes)
            } else {
```

```
                holder = (ViewHolder) convertView.getTag();
                holder.item = convertView;
            }
            holder.position = position;
            return holder;
        }

        @SuppressWarnings("unchecked")
        public <T extends View> T getView(int id) {
            T t = (T) mViews.get(id);
            if (t == null) {
                t = (T) item.findViewById(id);
                mViews.put(id, t);
            }
            return t;
        }

        /**
         * 获取当前条目
         */
        public View getItemView() {
            return item;
        }

        /**
         * 获取条目位置
         */
        public int getItemPosition() {
            return position;
        }

        /**
         * 设置文字
         */
        public ViewHolder setText(int id, CharSequence text) {
            View view = getView(id);
            if (view instanceof TextView) {
                ((TextView) view).setText(text);
            }
            return this;
        }

        /**
         * 设置图片
         */
        public ViewHolder setImageResource(int id, int drawableRes)
            View view = getView(id);
            if (view instanceof ImageView) {
                ((ImageView) view).setImageResource(drawableRes);
            } else {
                view.setBackgroundResource(drawableRes);
            }
            return this;
```

```
        }

        /**
         * 设置点击监听
         */
        public ViewHolder setOnClickListener(int id, View.OnClickLi
            getView(id).setOnClickListener(listener);
            return this;
        }

        /**
         * 设置可见
         */
        public ViewHolder setVisibility(int id, int visible) {
            getView(id).setVisibility(visible);
            return this;
        }

        /**
         * 设置标签
         */
        public ViewHolder setTag(int id, Object obj) {
            getView(id).setTag(obj);
            return this;
        }

        //其他方法可自行扩展

    }

}
```

本节小结：

> 本节给大家介绍了如何来实现一个可供复用的BaseAdapter，当然大家可以在这个的基础上根据 自己的需求进行修改，比如通过异步设置网络图片等~改代码是参考鸿洋大神的视频写的： 视频链接：Android-打造万能适配器 另外，实际编写中遇到一些问题，非常感谢**Berial(B神)**的耐心点拨~  ありがとうございます~

2.5.1 ListView Item多布局的实现

本节引言：

> 本节是ListView这个小节的最后一节，给大家带来的是ListView多布局Item的实现， 何为ListView Item多布局，打个比方，QQ这种聊天列表：
>
> 
>
>
> 假如他是用一个ListView做的，那么一个ListView上不就有两种不同的Item咯！ 一左一右，嘿嘿，本节就来教大家如何实现ListView的多布局！

1.要点讲解：

> 重写getItemViewType()方法对应View是哪个类别，以及getViewTypeCount()方法iew返回 总共多少个类别！然后再getView那里调用getItemViewType获得对应类别，再加载对应的View！

2.代码实现：

> 这里的话直接用上一节的两个布局，然后另外写一个Adapter重写要点中的几个几个地方：

MutiLayoutAdapter.java：

```
/**
 * Created by Jay on 2015/9/23 0023.
 */  public  class  MutiLayoutAdapter  extends  BaseAdapter{  //定
```

> 这里有个地方要注意的，convertView.setTag(R.id.Tag_APP,holder1);我们平时都直接 setTag(Object)的，这个是setTag的重载方法，参数是一个唯一的key以及后面的一个对象！唯一！！！我一开始直接把TYPE_BOOK作为第一个参数，然后就报下面这个错误：
>
> ```
> -23 14:09:21.280  30716-30716/? E/AndroidRuntime： FATAL EXCEPTION: main
>  Process: jay.com.listviewdemo6, PID: 30716
>  java.lang.IllegalArgumentException: The key must be an application-specific resource id.
> ```
>
> **The key must be an application-specific resource id** 就是前面这个要唯一，定义一个final类型的int变量和硬编码一个值的方式都是行不通的 这里的做法是直接在strings.xml中添加：

```
<item  name="Tag_APP"  type="id"></item>  <item  name="Tag_Book"  t
```

> 当然你也可以在res/values/下另外创建一个ids.xml文件，把上面这段代码贴上去！除了这个还有一个要注意的地方，就是这个区分类别的标志要从0开始算，不然会报下面 这样的错误：
>
> ```
> Process: jay.com.listviewdemo6, PID: 529
> java.lang.ArrayIndexOutOfBoundsException: length=2; index=2
>         at android.widget.AbsListView$RecycleBin.addScrapView(AbsListView.java:6563)
>         at android.widget.AbsListView.trackMotionScroll(AbsListView.java:4916)
>         at android.widget.AbsListView.scrollIfNeeded(AbsListView.java:3398)
> ```

MainActivity.java：

```
public  class  MainActivity  extends  AppCompatActivity  {  private
```

上面随机生成0和1，0就往集合中添加一个Book的对象，1的话就添加一个App的对象！

3.代码下载：

ListViewDemo6.zip

本节小结：

> 好的，本节给大家讲解了ListView Item多布局的实现，就是两个方法的重写，然后getView()做下判断，设置不同的布局而已~代码非常简单~
>
> 关于ListView的知识就告一段落吧，当然ListView的知识并不止这些， 异步加载， 优化等等，这些我们都会在进阶部分进行学习~就说这么多， 谢谢~

2.5.2 GridView(网格视图)的基本使用

本节引言：

> 本节给大家介绍的是第二个Adapter类的控件——GridView(网格视图)，见名知义，ListView是列表， GridView就是显示网格！他和ListView一样是AbsListView的子类！很多东西和ListView都是相通的， 本节我们就来学习他的基本用法~

1.相关属性：

下面是GridView中的一些属性：

> - **android:columnWidth**：设置列的宽度
> - **android:gravity**：组件对其方式
> - **android:horizontalSpacing**：水平方向每个单元格的间距
> - **android:verticalSpacing**：垂直方向每个单元格的间距
> - **android:numColumns**：设置列数
> - **android:stretchMode**：设置拉伸模式，可选值如下： **none**：不拉伸；**spacingWidth**：拉伸元素间的间隔空隙 **columnWidth**：仅仅拉伸表格元素自身 **spacingWidthUniform**：既拉元素间距又拉伸他们之间的间隔空袭

2.使用示例：

> 下面通过一个简单的例子来熟悉这个控件的使用：(这里用的Adapter我们直接用之2.5.0中教大家写的可复用的BaseAdapter~)

实现的效果图：



代码实现：

首先是GridView 的 Item的布局：**item_grid_icon.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp">

    <ImageView
        android:id="@+id/img_icon"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_centerInParent="true"
        android:src="@mipmap/iv_icon_1" />

    <TextView
        android:id="@+id/txt_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/img_icon"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:text="呵呵"
        android:textSize="18sp" />

</RelativeLayout>
```

接着我们写个entity实体类：**Icon.java**：

```
/**
 * Created by Jay on 2015/9/24 0024.
 */
public class Icon {
    private int iId;
    private String iName;

    public Icon() {
    }

    public Icon(int iId, String iName) {
        this.iId = iId;
        this.iName = iName;
    }

    public int getiId() {
        return iId;
    }

    public String getiName() {
        return iName;
    }

    public void setiId(int iId) {
        this.iId = iId;
    }

    public void setiName(String iName) {
        this.iName = iName;
    }
}
```

最后是MainActivity的布局以及Java代码

activity_main.xml：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    tools:context=".MainActivity">

    <!--numColumns设置每行显示多少个-->
    <GridView
        android:id="@+id/grid_photo"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:numColumns="3" />

</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private Context mContext;
    private GridView grid_photo;
    private BaseAdapter mAdapter = null;
    private ArrayList<Icon> mData = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        grid_photo = (GridView) findViewById(R.id.grid_photo);

        mData = new ArrayList<Icon>();
        mData.add(new Icon(R.mipmap.iv_icon_1, "图标1"));
        mData.add(new Icon(R.mipmap.iv_icon_2, "图标2"));
        mData.add(new Icon(R.mipmap.iv_icon_3, "图标3"));
        mData.add(new Icon(R.mipmap.iv_icon_4, "图标4"));
        mData.add(new Icon(R.mipmap.iv_icon_5, "图标5"));
        mData.add(new Icon(R.mipmap.iv_icon_6, "图标6"));
        mData.add(new Icon(R.mipmap.iv_icon_7, "图标7"));

        mAdapter = new MyAdapter<Icon>(mData, R.layout.item_grid_ic
            @Override
            public void bindView(ViewHolder holder, Icon obj) {
                holder.setImageResource(R.id.img_icon, obj.getiId(
                holder.setText(R.id.txt_icon, obj.getiName());
            }
        };

        grid_photo.setAdapter(mAdapter);

        grid_photo.setOnItemClickListener(new AdapterView.OnItemCl
            @Override
            public void onItemClick(AdapterView<?> parent, View vie
                Toast.makeText(mContext, "你点击了~" + position + "~
            }
        });

    }

}
```

嗯，代码非常简单~

3.示例代码下载：

GridViewDemo1.zip

本节小结：

> 本节给大家介绍了第二个需要使用Adapter的UI控件——网格视图GridView，用法很简单~ 大家可以根据自己的需求进行扩展，比如用GridView显示手机相册~嗯，就说这么多， 谢谢~

2.5.3 Spinner(列表选项框)的基本使用

本节引言：

> 本来本节是想给大家介绍一个Gallery(画廊)的一个控件的，后来想想还是算了，因为 在Android 4.1后就已经被弃用了，尽管我们可以通过兼容不来使用Gallery，不过想想 还是算了，因为Gallery在每次切换图片的时候，都需要重新创建视图，这样无疑会造成 很大资源浪费！我们可以通过其他方法来实现Gallery效果，比如通过HorizontalScrollView 来实现水平滚动效果，或者编写一个水平方向的ListView~有兴趣自己谷歌！
>
> 本节学习的是一个叫做Spinner的Adapter控件！ 应用场景：当我们的app需要用户输入数据时，除了让用户自己打字以外，还有一种比较贴心的设计: 列出一组选项让用户从中挑选，从而方便了我们的用户！ 话不多说，开始学习Spinner的基本用法~

1.相关属性

> - **android:dropDownHorizontalOffset**：设置列表框的水平偏移距离
> - **android:dropDownVerticalOffset**：设置列表框的水平竖直距离
> - **android:dropDownSelector**：列表框被选中时的背景
> - **android:dropDownWidth**：设置下拉列表框的宽度
> - **android:gravity**：设置里面组件的对其方式
> - **android:popupBackground**：设置列表框的背景
> - **android:prompt**：设置对话框模式的列表框的提示信息(标题)，只能够引用string.xml 中的资源id,而不能直接写字符串
> - **android:spinnerMode**：列表框的模式，有两个可选值： **dialog**：对话框风格的窗口 **dropdown**：下拉菜单风格的窗口(默认)
> - 可选属性：**android:entries**：使用数组资源设置下拉列表框的列表项目

2.使用示例：

> 对了，Spinner会默认选中第一个值，就是默认调用spinner.setSection(0), 你可以通过这个设置默认的选中值，另外，会触发一次OnItemSelectedListener 事件，暂时没找到解决方法，下面折衷的处理是：添加一个boolean值，然后设置 为false，在onItemSelected时进行判断，false说明是默认触发的，不做任何操作 将boolean值设置为true；true的话则正常触发事件！ 示例中写了两个不同的Spinner，从数据源，列表框风格等进行对比~ 接下来我们来看下

效果图：



代码实现：

这里依然使用的我们前面的可复用BaseAdapter：

第一个Spinner的数据源编写：

在**res/values**下编写一个：**myarrays.xml**的文件，内容如下：

```
<?xml version="1.0" encoding="utf-8"?>  <resources>  <string-array
```

接着是第二个Spinner的布局：**item_spin_hero.xml**：

```
<?xml version="1.0" encoding="utf-8"?>  <LinearLayout  xmlns:andro:
```

再接着编写一个Entitiy实体类：**Hero.java**：

```
/**
 * Created by Jay on 2015/9/24 0024.
 */  public  class  Hero  {  private  int hIcon;  private  String h
```

最后是MainActivity的布局与Java代码部分：

布局文件：**activity_main.xml**：

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

MainActivity.java：

```
public  class  MainActivity  extends  AppCompatActivity  implements
```

另外关于Spinner的OnItemSelectedListener，以及如何获得选中项的值，就自己看上面的 代码啦~

3.代码示例下载：

SpinnerDemo.zip

本节小结

> 好的，本节给大家介绍了Spinner(下拉选项框)的使用，例子还是蛮有趣的，哈哈~！ 别问我哪个区什么段位，我可是人机小王子，可惜一直在青铜分段苦苦挣扎~ 你知道为什么的，好吧，本节就到这里~

2.5.4 AutoCompleteTextView(自动完成文本框)的基本使用

本节引言：

> 本节继续来学习Adapter类的控件，这次带来的是AutoCompleteTextView(自动完成文本框)，相信细心的你发现了，和Adapter搭边的控件，都可以自己定义item的样式，是吧！或者说每个Item的布局~想怎么玩就怎么玩~嗯，话不多说，开始本节内容~ 对了贴下官方API：AutoCompleteTextView

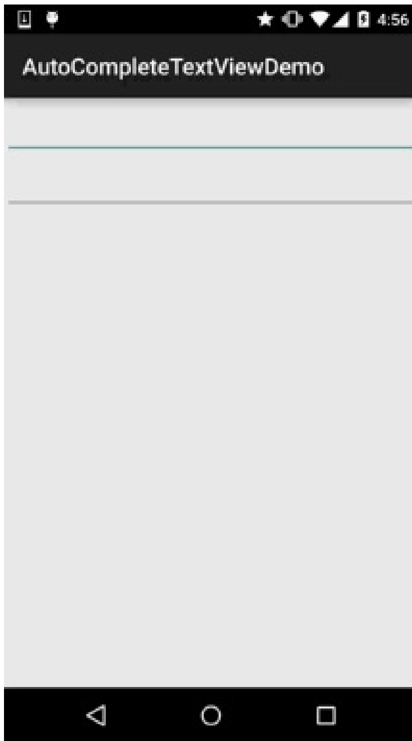
1.相关属性：

> - **android:completionHint**：设置下拉菜单中的提示标题
> - **android:completionHintView**：定义提示视图中显示下拉菜单
> - **android:completionThreshold**：指定用户至少输入多少个字符才会显示提示
> - **android:dropDownAnchor**：设置下拉菜单的定位"锚点"组件，如果没有指定改属性，将使用该TextView作为定位"锚点"组件
> - **android:dropDownHeight**：设置下拉菜单的高度
> - **android:dropDownWidth**：设置下拉菜单的宽度
> - **android:dropDownHorizontalOffset**：指定下拉菜单与文本之间的水平间距
> - **android:dropDownVerticalOffset**：指定下拉菜单与文本之间的竖直间距
> - **android:dropDownSelector**：设置下拉菜单点击效果
> - **android:popupBackground**：设置下拉菜单的背景

另外其实还有个**MultiAutoCompleteTextView**(多提示项的自动完成文本框) 和这个AutoCompleteTextView作用差不多，属性也一样，具体区别在哪里，我们在下面的代码中来体验~另外这两个都是全词匹配的，比如，小猪猪：你输入小->会提示小猪猪，但是输入猪猪->却不会提示小猪猪！

2.代码示例：

运行效果图：



实现代码：

这里的话就不自定义布局了，直接用ArrayAdapter来实现吧！

布局文件：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <AutoCompleteTextView
        android:id="@+id/atv_content"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:completionHint="请输入搜索内容"
        android:completionThreshold="1"
        android:dropDownHorizontalOffset="5dp" />

    <MultiAutoCompleteTextView
        android:id="@+id/matv_content"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:completionThreshold="1"
        android:dropDownHorizontalOffset="5dp"
        android:text="" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private AutoCompleteTextView atv_content;
    private MultiAutoCompleteTextView matv_content;

    private static final String[] data = new String[]{
            "小猪猪", "小狗狗", "小鸡鸡", "小猫猫", "小咪咪"
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        atv_content = (AutoCompleteTextView) findViewById(R.id.atv_
        matv_content = (MultiAutoCompleteTextView) findViewById(R.i

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(Ma
                this, android.R.layout.simple_dropdown_item_1line,
        atv_content.setAdapter(adapter);

        ArrayAdapter<String> adapter2 = new ArrayAdapter<String>(ge
        matv_content.setAdapter(adapter);
        matv_content.setTokenizer(new MultiAutoCompleteTextView.Com
    }
}
```

部分代码分析：

1. android:completionThreshold="1"：这里我们设置了输入一个字就显示提示
2. android:completionHint="请输入搜索内容"：这是框框底部显示的文字，如果觉得丑 可以android:completionHintView设置一个View!
3. android:dropDownHorizontalOffset="5dp"：设置了水平边距为5dp
4. matv_content.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer()); setTokenizer是为其设置分隔符

3.示例代码下载：

AutoCompleteTextViewDemo.zip

本节小结：

> 本节给大家介绍了AutoCompleteTextView(自动完成文本框)，非常简单~ 大家可根据实际开发需求自行拓展~好的，就说这么多，谢谢~

2.5.5 ExpandableListView(可折叠列表)的基本使用

本节引言：

> 本节要讲解的Adapter类控件是ExpandableListView，就是可折叠的列表，它是ListView的子类，在ListView的基础上它把应用中的列表项分为几组，每组里又可包含多个列表项。至于样子，类似于QQ联系人列表，他的用法与ListView非常相似，只是ExpandableListVivew显示的列表项 需由ExpandableAdapter提供。下面我们来学习这个控件的基本使用！官方API：ExpandableListView

1.相关属性

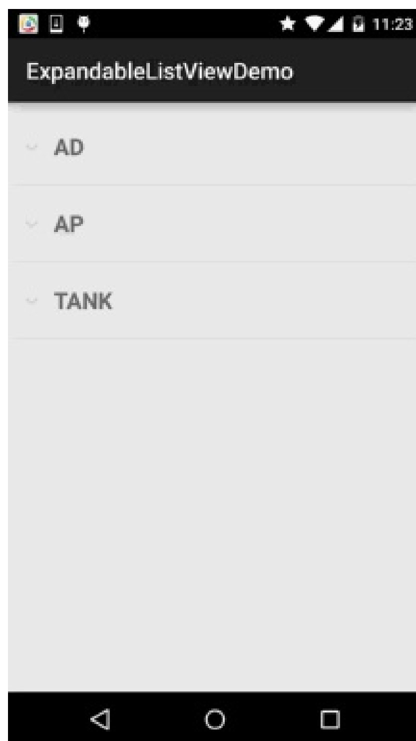
- **android:childDivider**：指定各组内子类表项之间的分隔条，图片不会完全显示，分离子列表项的是一条直线
- **android:childIndicator**：显示在子列表旁边的Drawable对象，可以是一个图像
- **android:childIndicatorEnd**：子列表项指示符的结束约束位置
- **android:childIndicatorLeft**：子列表项指示符的左边约束位置
- **android:childIndicatorRight**：子列表项指示符的右边约束位置
- **android:childIndicatorStart**：子列表项指示符的开始约束位置
- **android:groupIndicator**：显示在组列表旁边的Drawable对象，可以是一个图像
- **android:indicatorEnd**：组列表项指示器的结束约束位置
- **android:indicatorLeft**：组列表项指示器的左边约束位置
- **android:indicatorRight**：组列表项指示器的右边约束位置
- **android:indicatorStart**：组列表项指示器的开始约束位置

2.实现ExpandableAdapter的三种方式

> **1.** 扩展**BaseExpandableListAdpter**实现ExpandableAdapter。
>
> **2.** 使用**SimpleExpandableListAdpater**将两个List集合包装成ExpandableAdapter
>
> **3.** 使用**simpleCursorTreeAdapter**将Cursor中的数据包装成SimpleCuroTreeAdapter 本节示例使用的是第一个，扩展BaseExpandableListAdpter，我们需要重写该类中的相关方法，下面我们通过一个代码示例来体验下！

3.代码示例

我们来看下实现的效果图：



下面我们就来实现上图的这个效果：

核心是重写**BaseExpandableListAdpter**，其实和之前写的普通的BaseAdapter是类似的， 但是BaseExpandableListAdpter则分成了两部分：组和子列表，具体看代码你就知道了！

另外，有一点要注意的是，重写**isChildSelectable()**方法需要返回true，不然不会触发 子Item的点击事件！下面我们来写写：

首先是组和子列表的布局：

item_exlist_group.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="5dp">

    <TextView
        android:id="@+id/tv_group_name"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:gravity="center_vertical"
        android:paddingLeft="30dp"
        android:text="AP"
        android:textStyle="bold"
        android:textSize="20sp" />

</LinearLayout>
```

item_exlist_item.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="5dp"
    android:background="#6BBA79">

    <ImageView
        android:id="@+id/img_icon"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:src="@mipmap/iv_lol_icon1"
        android:focusable="false"/>

    <TextView
        android:id="@+id/tv_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:layout_marginTop="15dp"
        android:focusable="false"
        android:text="提莫"
        android:textSize="18sp" />

</LinearLayout>
```

然后是自定义的Adapter类：

MyBaseExpandableListAdapter.java：

```
/**
 * Created by Jay on 2015/9/25 0025.
 */
public class MyBaseExpandableListAdapter extends BaseExpandableList

    private ArrayList<Group> gData;
    private ArrayList<ArrayList<Item>> iData;
    private Context mContext;

    public MyBaseExpandableListAdapter(ArrayList<Group> gData,Array
        this.gData = gData;
        this.iData = iData;
        this.mContext = mContext;
    }

    @Override
    public int getGroupCount() {
        return gData.size();
    }

    @Override
    public int getChildrenCount(int groupPosition) {
        return iData.get(groupPosition).size();
    }

    @Override
    public Group getGroup(int groupPosition) {
        return gData.get(groupPosition);
    }

    @Override
    public Item getChild(int groupPosition, int childPosition) {
        return iData.get(groupPosition).get(childPosition);
    }

    @Override
    public long getGroupId(int groupPosition) {
        return groupPosition;
    }

    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return childPosition;
    }

    @Override
    public boolean hasStableIds() {
        return false;
```

```
    }

    //取得用于显示给定分组的视图. 这个方法仅返回分组的视图对象
    @Override
    public View getGroupView(int groupPosition, boolean isExpanded,

        ViewHolderGroup groupHolder;
        if(convertView == null){
            convertView = LayoutInflater.from(mContext).inflate(
                    R.layout.item_exlist_group, parent, false);
            groupHolder = new ViewHolderGroup();
            groupHolder.tv_group_name = (TextView) convertView.finc
            convertView.setTag(groupHolder);
        }else{
            groupHolder = (ViewHolderGroup) convertView.getTag();
        }
        groupHolder.tv_group_name.setText(gData.get(groupPosition).
        return convertView;
    }

    //取得显示给定分组给定子位置的数据用的视图
    @Override
    public View getChildView(int groupPosition, int childPosition,
        ViewHolderItem itemHolder;
        if(convertView == null){
            convertView = LayoutInflater.from(mContext).inflate(
                    R.layout.item_exlist_item, parent, false);
            itemHolder = new ViewHolderItem();
            itemHolder.img_icon = (ImageView) convertView.findViewE
            itemHolder.tv_name = (TextView) convertView.findViewByI
            convertView.setTag(itemHolder);
        }else{
            itemHolder = (ViewHolderItem) convertView.getTag();
        }
        itemHolder.img_icon.setImageResource(iData.get(groupPositic
        itemHolder.tv_name.setText(iData.get(groupPosition).get(chi
        return convertView;
    }

    //设置子列表是否可选中
    @Override
    public boolean isChildSelectable(int groupPosition, int childPc
        return true;
    }

    private static class ViewHolderGroup{
        private TextView tv_group_name;
    }

    private static class ViewHolderItem{
        private ImageView img_icon;
        private TextView tv_name;
    }
```

```
}
```

PS：存储子列表的数据不一定要用ArrayList<ArrayList<item>>这种，根据自己的需求 定义~</item>

最后是MainActivity的布局以及Java代码：

布局文件:**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    tools:context=".MainActivity">

    <ExpandableListView
        android:id="@+id/exlist_lol"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:childDivider="#E02D2F"/>

</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private ArrayList<Group> gData = null;
    private ArrayList<ArrayList<Item>> iData = null;
    private ArrayList<Item> lData = null;
    private Context mContext;
    private ExpandableListView exlist_lol;
    private MyBaseExpandableListAdapter myAdapter = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        exlist_lol = (ExpandableListView) findViewById(R.id.exlist_

        //数据准备
        gData = new ArrayList<Group>();
        iData = new ArrayList<ArrayList<Item>>();
        gData.add(new Group("AD"));
        gData.add(new Group("AP"));
```

```
        gData.add(new Group("TANK"));

        lData = new ArrayList<Item>();

        //AD组
        lData.add(new Item(R.mipmap.iv_lol_icon3,"剑圣"));
        lData.add(new Item(R.mipmap.iv_lol_icon4,"德莱文"));
        lData.add(new Item(R.mipmap.iv_lol_icon13,"男枪"));
        lData.add(new Item(R.mipmap.iv_lol_icon14,"韦鲁斯"));
        iData.add(lData);
        //AP组
        lData = new ArrayList<Item>();
        lData.add(new Item(R.mipmap.iv_lol_icon1, "提莫"));
        lData.add(new Item(R.mipmap.iv_lol_icon7, "安妮"));
        lData.add(new Item(R.mipmap.iv_lol_icon8, "天使"));
        lData.add(new Item(R.mipmap.iv_lol_icon9, "泽拉斯"));
        lData.add(new Item(R.mipmap.iv_lol_icon11, "狐狸"));
        iData.add(lData);
        //TANK组
        lData = new ArrayList<Item>();
        lData.add(new Item(R.mipmap.iv_lol_icon2, "诺手"));
        lData.add(new Item(R.mipmap.iv_lol_icon5, "德邦"));
        lData.add(new Item(R.mipmap.iv_lol_icon6, "奥拉夫"));
        lData.add(new Item(R.mipmap.iv_lol_icon10, "龙女"));
        lData.add(new Item(R.mipmap.iv_lol_icon12, "狗熊"));
        iData.add(lData);

        myAdapter = new MyBaseExpandableListAdapter(gData,iData,mCo
        exlist_lol.setAdapter(myAdapter);

        //为列表设置点击事件
        exlist_lol.setOnChildClickListener(new ExpandableListView.O
            @Override
            public boolean onChildClick(ExpandableListView parent,
                Toast.makeText(mContext, "你点击了：" + iData.get(gr
                return true;
            }
        });

    }
}
```

4.代码下载：

ExpandableListViewDemo.zip

本节小结：

> 好的，本节给大家介绍了ExpandableListView的基本使用，嘿嘿，有点意思~
> 这里只是一个示例，其他的根据自己的需求自行扩展~谢谢

2.5.6 ViewFlipper(翻转视图)的基本使用

本节引言：

> 本节给大家带了的是ViewFlipper，它是Android自带的一个多页面管理控件，且可以自动播放！和ViewPager不同，ViewPager是一页页的，而ViewFlipper则是一层层的，和ViewPager一样，很多时候，用来实现进入应用后的引导页，或者用于图片轮播，本节我们就使用ViewFlipper写一个简单的图片轮播的例子吧~官方API：ViewFlipper

1.为ViewFlipper加入View的两种方法

1）静态导入

> 所谓的静态导入就是像图中这样，把个个页面添加到ViewFlipper的中间！

```
<ViewFlipper
    android:id="@+id/vflp_help"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <include layout="@layout/page_help_one" />

    <include layout="@layout/page_help_two" />

    <include layout="@layout/page_help_three" />

    <include layout="@layout/page_help_four" />

</ViewFlipper>
```

2）动态导入

> 通过addView方法填充View

```
vflp_help = (ViewFlipper) findViewById(R.id.vflp_help);
vflp_help.addView(v1);
vflp_help.addView(v2);
vflp_help.addView(v3);
vflp_help.addView(v4);
```

2.常用的一些方法

> - **setInAnimation**：设置View进入屏幕时使用的动画
> - **setOutAnimation**：设置View退出屏幕时使用的动画
> - **showNext**：调用该方法来显示ViewFlipper里的下一个View
> - **showPrevious**：调用该方法来显示ViewFlipper的上一个View
> - **setFilpInterval**：设置View之间切换的时间间隔
> - **setFlipping**：使用上面设置的时间间隔来开始切换所有的View，切换会循环进行
> - **stopFlipping**：停止View切换

3.使用实例

1）示例1：使用ViewFlipper实现图片轮播(静态导入)

实现效果图：



实现代码：

每个页面的布局都是一个简单的ImageView，这里就不贴了~先贴下两个进入以及离开的动画：

right_in.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate
        android:duration="2000"
        android:fromXDelta="100%p"
        android:toXDelta="0" />

</set>
```

right_out.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >

    <translate
        android:duration="2000"
        android:fromXDelta="0"
        android:toXDelta="-100%p" />

</set>
```

然后是**activity_main.xml**布局文件：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ViewFlipper
        android:id="@+id/vflp_help"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:inAnimation="@anim/right_in"
        android:outAnimation="@anim/right_out"
        android:flipInterval="3000">

        <include layout="@layout/page_help_one" />

        <include layout="@layout/page_help_two" />

        <include layout="@layout/page_help_three" />

        <include layout="@layout/page_help_four" />

    </ViewFlipper>

</RelativeLayout>
```

这里我们设置了flipInterval = 3000，即每隔3000ms切还一个~ 最后我们只需在MainActivity.java中调用ViewFlipper的startFlipping()方法开始滑动！

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private ViewFlipper vflp_help;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        vflp_help = (ViewFlipper) findViewById(R.id.vflp_help);
        vflp_help.startFlipping();
    }
}
```

2）示例2：支持手势滑动的ViewFlipper(动态导入)

实现效果图：



代码实现：

因为我们分为进入上一页，进入下一页，所以除了上面的两个动画外，我们再添加两个动画：

left_in.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >

    <translate
        android:duration="500"
        android:fromXDelta="-100%p"
        android:toXDelta="0" />

</set>
```

left_out.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate
        android:duration="500"
        android:fromXDelta="0"
        android:toXDelta="100%p" />

</set>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity  {

    private Context mContext;
    private ViewFlipper vflp_help;
    private int[] resId = {R.mipmap.ic_help_view_1,R.mipmap.ic_help
            R.mipmap.ic_help_view_3,R.mipmap.ic_help_view_4};

    private final static int MIN_MOVE = 200;   //最小距离
    private MyGestureListener mgListener;
    private GestureDetector mDetector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        //实例化SimpleOnGestureListener与GestureDetector对象
        mgListener = new MyGestureListener();
        mDetector = new GestureDetector(this, mgListener);
        vflp_help = (ViewFlipper) findViewById(R.id.vflp_help);
        //动态导入添加子View
        for(int i = 0;i < resId.length;i++){
            vflp_help.addView(getImageView(resId[i]));
        }

    }

    //重写onTouchEvent触发MyGestureListener里的方法
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return mDetector.onTouchEvent(event);
    }

    //自定义一个GestureListener,这个是View类下的，别写错哦！！！
    private class MyGestureListener extends GestureDetector.Simple
        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, floa
            if(e1.getX() - e2.getX() > MIN_MOVE){
                vflp_help.setInAnimation(mContext,R.anim.right_in)
                vflp_help.setOutAnimation(mContext, R.anim.right_ou
                vflp_help.showNext();
            }else if(e2.getX() - e1.getX() > MIN_MOVE){
                vflp_help.setInAnimation(mContext,R.anim.left_in);
                vflp_help.setOutAnimation(mContext, R.anim.left_out
                vflp_help.showPrevious();
            }
            return true;
        }
    }
```

```
    private ImageView getImageView(int resId){
        ImageView img = new ImageView(this);
        img.setBackgroundResource(resId);
        return img;
    }
}
```

代码要点解析：

> 1.这里我们通过动态的方法添加View，这里只是简单的ImageView，可根据自己需求进行扩展！2.相信细心的你发现了，这里我们的手势用的不是通过onTouchEvent直接判断的，然后重写 onTouch事件，对Action进行判断，然后如果是MotionEvent.ACTION_MOVE的话，就执行下述代码：
>
> ```
> if(event.getX() > startX){
>     vflp_help.setInAnimation(this,R.anim.left_in);
>     vflp_help.setOutAnimation(this, R.anim.left_out);
>     vflp_help.showPrevious();
> }else if(startX > event.getX()){
>     vflp_help.setInAnimation(this,R.anim.right_in);
>     vflp_help.setOutAnimation(this, R.anim.right_out);
>     vflp_help.showNext();
> }
> ```
>
> 后来发现，模拟器上因为是鼠标的关系，并不会频繁的抖动，而真机上，因为手指一直是颤抖的 所以ACTION_MOVE会一直触发，然后View一直切换，后来考虑了**Berial(B神)**的建议，加入了 一个值来进行判断，就是添加一个标志：
>
> ```
> switch (event.getAction()){
>     case MotionEvent.ACTION_DOWN:
>         startX = event.getX();
>         isChange = true;
>         break;
>     case MotionEvent.ACTION_MOVE:
>         if(isChange){
>             //向右滑动，看前一页
>             if(event.getX() > startX){
>                 vflp_help.setInAnimation(this,R.anim.left_in);
>                 vflp_help.setOutAnimation(this, R.anim.left_out);
>                 vflp_help.showPrevious();
>             }else if(startX > event.getX()){
>                 vflp_help.setInAnimation(this,R.anim.right_in);
>                 vflp_help.setOutAnimation(this, R.anim.right_out);
>                 vflp_help.showNext();
>             }
>             isChange = false;
>         }
>
>         break;
>     case MotionEvent.ACTION_UP:
> ```
>
> 可以是可以，不过感觉还是有点不流畅，怪怪的，后来想想还是用手势类，直接在onFling处理 就好，于是就有了上面的代码，运行起来杠杠滴~ 当然，如果你对Gesture手势不熟悉的话，可以参见之前写过的一篇文章： Android基础入门教程——3.8 Gesture(手势)

4.代码示例下载

ViewFlipperDemo.zip

ViewFlipperDemo2.zip

本节小结：

> 好的，本节给大家讲解了ViewFlipper(翻转视图)的基本使用，以后做图片轮播和引导页， 你就多了一个选择了~嗯，就说这么多，谢谢~

2.5.7 Toast(吐司)的基本使用

本节引言：

> 好的，终于学习完Adapter类相关的一些控件，当然除了讲解的那几个，还有其他很多的 相关的控件，就不慢慢讲解了~有需要的自行查阅文档，查看相关的用法，本节带来的是： Android用于提示信息的一个控件——Toast(吐司)！Toast是一种很方便的消息提示框,会在 屏幕中显示一个消息提示框,没任何按钮,也不会获得焦点一段时间过后自动消失！ 非常常用！本节我们就来学习Toast的使用！

1.直接调用Toast类的makeText()方法创建

这是我们用的最多的一种形式了！比如点击一个按钮，然后弹出Toast，用法：**Toast.makeText(MainActivity.this, "提示的内容", Toast.LENGTH_LONG).show();** 第一个是上下文对象！对二个是显示的内容！第三个是显示的时间，只有LONG和SHORT两种 会生效，即时你定义了其他的值，最后调用的还是这两个！

另外Toast是非常常用的，我们可以把这些公共的部分抽取出来，写到一个方法里！需要显示Toast的时候直接调用这个方法就可以显示Toast，这样方便很多！ 示例如下：

>

```
void midToast(String str, int showTime)
{
    Toast toast = Toast.makeText(global_context, str, showTime);
    toast.setGravity(Gravity.CENTER_VERTICAL|Gravity.CENTER_HORIZON
    TextView v = (TextView) toast.getView().findViewById(android.R.
    v.setTextColor(Color.YELLOW);     //设置字体颜色
    toast.show();
}
```

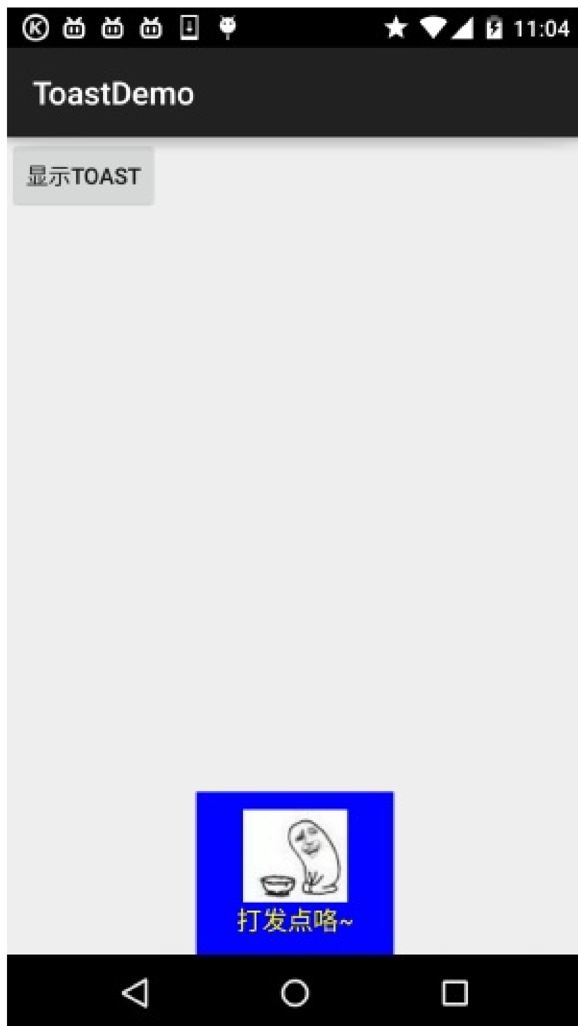
上面这个抽取出来的方法，我们发现我们可以调用setGravity设置Toast显示的位置以及获得 通过findViewById(android.R.id.message)获得显示的文本，然后进行设置颜色，或者大小等！ 这就是第二种通过构造方法来定制Toast!

2.通过构造方法来定制Toast：

上面定制了文本，以及显示位置，下面我们写两个简单的例子：

1.定义一个带有图片的Toast

效果图：



关键代码：

```
private void midToast(String str, int showTime)
{
    Toast toast = Toast.makeText(mContext, str, showTime);
    toast.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.BOTTOM , 0,
    LinearLayout layout = (LinearLayout) toast.getView();
    layout.setBackgroundColor(Color.BLUE);
    ImageView image = new ImageView(this);
    image.setImageResource(R.mipmap.ic_icon_qitao);
    layout.addView(image, 0);
    TextView v = (TextView) toast.getView().findViewById(android.R.
    v.setTextColor(Color.YELLOW);     //设置字体颜色
    toast.show();
}
```

2.Toast完全自定义

> 如果上面的那种还满足不了你的话，那么你完全可以自己写一个Toast的布局，然后显示出来；但是时间我们依旧控制不了！

运行效果图：



关键代码：

```
private void midToast(String str, int showTime)
{
    LayoutInflater inflater = getLayoutInflater();
    View view = inflater.inflate(R.layout.view_toast_custom,
            (ViewGroup) findViewById(R.id.lly_toast));
    ImageView img_logo = (ImageView) view.findViewById(R.id.img_lo
    TextView tv_msg = (TextView) view.findViewById(R.id.tv_msg);
    tv_msg.setText(str);
    Toast toast = new Toast(mContext);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.setDuration(Toast.LENGTH_LONG);
    toast.setView(view);
    toast.show();
}
```

还有自定义Toast的布局以及圆角背景：

圆角背景：**bg_toast.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 设置透明背景色 -->
    <solid android:color="#BADB66" />
    <!-- 设置一个黑色边框 -->
    <stroke
        android:width="1px"
        android:color="#FFFFFF" />
    <!-- 设置四个圆角的半径 -->
    <corners
        android:bottomLeftRadius="50px"
        android:bottomRightRadius="50px"
        android:topLeftRadius="50px"
        android:topRightRadius="50px" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:bottom="5dp"
        android:left="5dp"
        android:right="5dp"
        android:top="5dp" />
</shape>
```

布局文件：**view_toast_custom.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:id="@+id/lly_toast"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg_toast"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/img_logo"
        android:layout_width="24dp"
        android:layout_height="24dp"
        android:layout_marginLeft="10dp"
        android:src="@mipmap/iv_lol_icon1" />

    <TextView
        android:id="@+id/tv_msg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:textSize="20sp" />

</LinearLayout>
```

非常简单，嘿嘿~

3.示例代码下载

ToastDemo.zip

本节小结：

好的，本节给大家讲解了Toast的基本使用，以及如何自定义Toast，非常简单，大家可以在实际开发中对自己的Toast进行定制~

2.5.8 Notification(状态栏通知)详解

本节引言：

> 本节带来的是Android中用于在状态栏显示通知信息的控件：Notification，相信大部分 学Android都对他都很熟悉，而网上很多关于Notification的使用教程都是基于2.x的，而 现在普遍的Android设备基本都在4.x以上，甚至是5.0以上的都有；他们各自的Notification 都是不一样的！而本节给大家讲解的是基于4.x以上的Notification，而5.0以上的Notification 我们会在进阶教程的Android 5.0新特性的章节进行讲解~
>
> 官方文档对Notification的一些介绍：
>
> 设计思想：Notifications in Android 4.4 and Lower
>
> 译文：通知
>
> **API**文档：Notification
>
> 访问上述网站，可能需要梯子哦~

1.设计文档部分解读

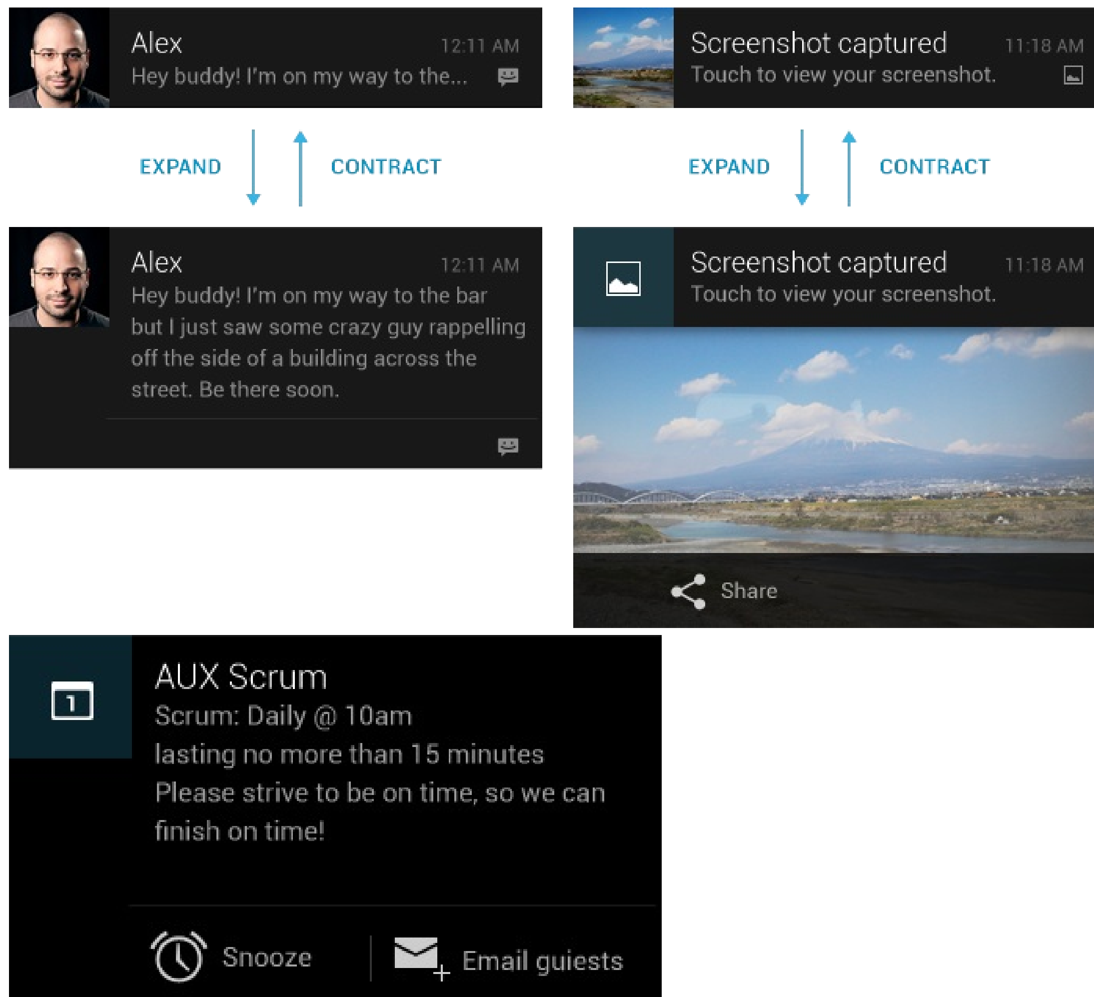
1）Notification的基本布局



> 上面的组成元素依次是：
>
> - **Icon/Photo**：大图标
> - **Title/Name**：标题
> - **Message**：内容信息
> - **Timestamp**：通知时间，默认是系统发出通知的时间，也可以通过setWhen()来设置
> - **Secondary Icon**：小图标
> - 内容文字，在小图标的左手边的一个文字

2）扩展布局

在 Jelly Bean 中你可以为通知提供更多事件的细节。你可以通过扩展布局显示消息的前几行或者图片的预览。这样用户可以看多更多的内容 - 有时甚至可以看到整个消息。用户可以通过 pinch-zoom 或者双手指滑动来打开扩展布局。Android 为单条消息提供了两种扩展布局 (文字和图像) 供你开发应用时使用。



关于其他一些设计的东西，就不一一提及了，有兴趣的自行查看上面提供的API文档，知道下 这个Notification在4.x以上的版本可以多种多样就好！我们更多的时候关注的是 如何写代码使用这个东西，下面我们就来学习下Notification的用法！

2.Notification的基本使用流程

> 状态通知栏主要涉及到2个类：Notification 和NotificationManager
>
> **Notification**：通知信息类，它里面对应了通知栏的各个属性
>
> **NotificationManager**：是状态栏通知的管理类，负责发通知、清除通知等操作。
>
> 使用的基本流程：
>
> - **Step 1.** 获得NotificationManager对象： NotificationManager mNManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
> - **Step 2.** 创建一个通知栏的Builder构造类： Notification.Builder mBuilder = new Notification.Builder(this);
> - **Step 3.** 对Builder进行相关的设置，比如标题，内容，图标，动作等！
> - **Step 4.**调用Builder的build()方法为notification赋值
> - **Step 5**.调用NotificationManager的notify()方法发送通知！
> - **PS:**另外我们还可以调用NotificationManager的cancel()方法取消通知

3.设置相关的一些方法：

> Notification.Builder mBuilder = new Notification.Builder(this);
>
> 后再调用下述的相关的方法进行设置：(官方API文档：Notification.Builder) 常用的方法如下：
>
> - **setContentTitle**(CharSequence)：设置标题
> - **setContentText**(CharSequence)：设置内容
> - **setSubText**(CharSequence)：设置内容下面一小行的文字
> - **setTicker**(CharSequence)：设置收到通知时在顶部显示的文字信息
> - **setWhen**(long)：设置通知时间，一般设置的是收到通知时的System.currentTimeMillis()
> - **setSmallIcon**(int)：设置右下角的小图标，在接收到通知的时候顶部也会显示这个小图标
> - **setLargeIcon**(Bitmap)：设置左边的大图标
> - **setAutoCancel**(boolean)：用户点击Notification点击面板后是否让通知取消(默认不取消)
> - **setDefaults**(int)：向通知添加声音、闪灯和振动效果的最简单、 使用默认（defaults）属性，可以组合多个属性，**Notification.DEFAULT_VIBRATE**(添加默认震动提醒)；**Notification.DEFAULT_SOUND**(添加默认声音提醒)；**Notification.DEFAULT_LIGHTS**(添加默认三色灯提醒) **Notification.DEFAULT_ALL**(添加默认以上3种全部提醒)
> - **setVibrate**(long[])：设置振动方式，比如： setVibrate(new long[] {0,300,500,700});延迟0ms，然后振动300ms，在延迟500ms， 接着再振动700ms，关于Vibrate用法后面会讲解！
> - **setLights**(int argb, int onMs, int offMs)：设置三色灯，参数依次是：灯光颜色， 亮持续时间，暗的时间，不是所有颜色都可以，这跟设备有关，有些手机还不带三色灯； 另外，还需要为Notification设置flags为Notification.FLAG_SHOW_LIGHTS才支持三色灯提醒！

- **setSound**(Uri)：设置接收到通知时的铃声，可以用系统的，也可以自己设置，例子如下: .setDefaults(Notification.DEFAULT_SOUND) //获取默认铃声 .setSound(Uri.parse("file:///sdcard/xx/xx.mp3")) //获取自定义铃声 .setSound(Uri.withAppendedPath(Audio.Media.INTERNAL_CONTENT_URI, "5")) //获取Android多媒体库内的铃声

- **setOngoing**(boolean)：设置为ture，表示它为一个正在进行的通知。他们通常是用来表示 一个后台任务,用户积极参与(如播放音乐)或以某种方式正在等待,因此占用设备(如一个文件下载, 同步操作,主动网络连接)

- **setProgress**(int,int,boolean)：设置带进度条的通知 参数依次为：进度条最大数值，当前进度，进度是否不确定 如果为确定的进度条：调用setProgress(max, progress, false)来设置通知， 在更新进度的时候在此发起通知更新progress，并且在下载完成后要移除进度条，通过调用setProgress(0, 0, false)既可。如果为不确定（持续活动）的进度条， 这是在处理进度无法准确获知时显示活动正在持续，所以调用setProgress(0, 0, true)，操作结束时，调用setProgress(0, 0, false)并更新通知以移除指示条
- **setContentIntent**(PendingIntent)：PendingIntent和Intent略有不同，它可以设置执行次数， 主要用于远程服务通信、闹铃、通知、启动器、短信中，在一般情况下用的比较少。比如这里通过 Pending启动Activity：getActivity(Context, int, Intent, int)，当然还可以启动Service或者Broadcast PendingIntent的位标识符(第四个参数)： **FLAG_ONE_SHOT** 表示返回的PendingIntent仅能执行一次，执行完后自动取消 **FLAG_NO_CREATE** 表示如果描述的PendingIntent不存在，并不创建相应的PendingIntent，而是返回NULL **FLAG_CANCEL_CURRENT** 表示相应的PendingIntent已经存在，则取消前者，然后创建新的PendingIntent，这个有利于数据保持为最新的，可以用于即时通信的通信场景 **FLAG_UPDATE_CURRENT** 表示更新的PendingIntent 使用示例：

```
```
//点击后跳转Activity
Intent intent = new Intent(context,XXX.class);
PendingIntent pendingIntent = PendingIntent.getActivity(context
mBuilder.setContentIntent(pendingIntent)

```
```

- **setPriority**(int)：设置优先级：

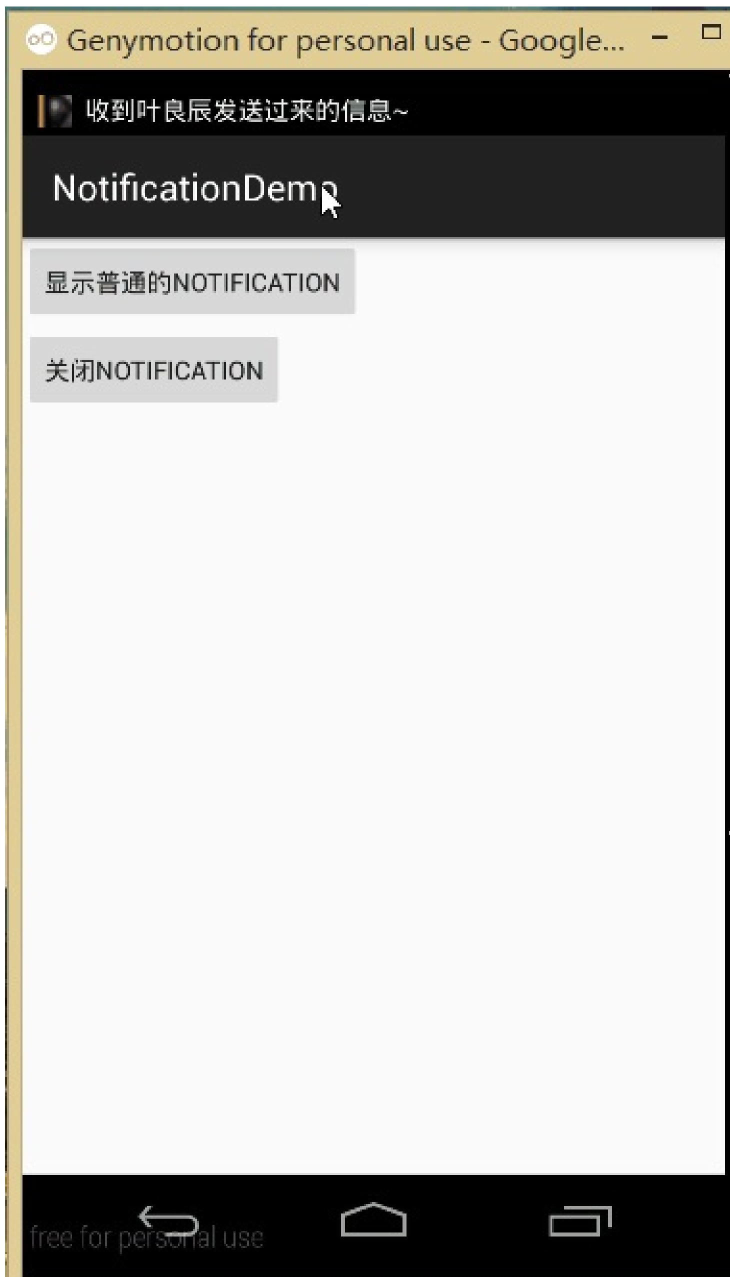
```
| 优先级 | 用户 |
| :-- | :-- |
| MAX | 重要而紧急的通知，通知用户这个事件是时间上紧迫的或者需要立即处理的
| HIGH | 高优先级用于重要的通信内容，例如短消息或者聊天，这些都是对用户来
| DEFAULT | 默认优先级用于没有特殊优先级分类的通知。 |
| LOW | 低优先级可以通知用户但又不是很紧急的事件。 |
| MIN | 用于后台消息 (例如天气或者位置信息)。最低优先级通知将只在状态栏显

对应属性：Notification.PRIORITY_HIGH...
```

4.代码示例：最常见的Notification：

下面我们来写一个最简单的例子来体验下Notification的用法：

运行效果图：



关键代码：

这里直接贴**MainActivity.java**的代码：

```
public class MainActivity extends AppCompatActivity implements View

    private Context mContext;
    private NotificationManager mNManager;
    private Notification notify1;
    Bitmap LargeBitmap = null;
    private static final int NOTIFYID_1 = 1;

    private Button btn_show_normal;
    private Button btn_close_normal;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        //创建大图标的Bitmap
        LargeBitmap = BitmapFactory.decodeResource(getResources(),
        mNManager = (NotificationManager) getSystemService(NOTIFICA
        bindView();

    }

    private void bindView() {
        btn_show_normal = (Button) findViewById(R.id.btn_show_norma
        btn_close_normal = (Button) findViewById(R.id.btn_close_nor
        btn_show_normal.setOnClickListener(this);
        btn_close_normal.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_show_normal:
                //定义一个PendingIntent点击Notification后启动一个Activ
                Intent it = new Intent(mContext, OtherActivity.clas
                PendingIntent pit = PendingIntent.getActivity(mCont

                //设置图片,通知标题,发送时间,提示方式等属性
                Notification.Builder mBuilder = new Notification.Bu
                mBuilder.setContentTitle("叶良辰")
                        .setContentText("我有一百种方法让你呆不下去~")
                        .setSubText("——记住我叫叶良辰")
                        .setTicker("收到叶良辰发送过来的信息~")
                        .setWhen(System.currentTimeMillis())
                        .setSmallIcon(R.mipmap.ic_lol_icon)
                        .setLargeIcon(LargeBitmap)
                        .setDefaults(Notification.DEFAULT_LIGHTS |
                        .setSound(Uri.parse("android.resource://" +
                        .setAutoCancel(true)
                        .setContentIntent(pit);
                notify1 = mBuilder.build();
                mNManager.notify(NOTIFYID_1, notify1);
                break;

            case R.id.btn_close_normal:
                //除了可以根据ID来取消Notification外,还可以调用cancelAl
                mNManager.cancel(NOTIFYID_1);
                break;

        }
    }
}
```

注释很详细，就不一一细讲了~

5.代码示例下载：

NotificationDemo.zip

本节小结：

> 好的，本节给大家介绍了Notification在4.x版本的基本用法，非常简单是吧~ 当然你也可以自定义Notification有兴趣的可以自己查阅相关资料，这里就不慢慢研究了~对了，本节部分内容参考的下述blog，贴下链接，大家也可以去看下：Android 通知栏Notification的整合 全面学习 （一个DEMO让你完全了解它） 写得蛮详细的~本节就到这里，谢谢~

2.5.9 AlertDialog(对话框)详解

本节引言：

> 本节继续给大家带来是显示提示信息的第三个控件AlertDialog(对话框)，同时它也是其他 Dialog的的父类！比如ProgressDialog，TimePickerDialog等，而AlertDialog的父类是：Dialog！另外，不像前面学习的Toast和Notification，AlertDialog并不能直接new出来，如果你打开 AlertDialog的源码，会发现构造方法是protected的，如果我们要创建AlertDialog的话，我们需要使用到该类中的一个静态内部类：public static class **Builder**，然后来调用AlertDialog 里的相关方法，来对AlertDialog进行定制，最后调用show()方法来显示我们的AlertDialog对话框！好的，下面我们就来学习AlertDialog的基本用法，以及定制我们的AlertDialog！官方文档：AlertDialog

1.基本使用流程

> - **Step 1**：创建**AlertDialog.Builder**对象；
> - **Step 2**：调用**setIcon()**设置图标，**setTitle()**或**setCustomTitle()**设置标题；
> - **Step 3**：设置对话框的内容：**setMessage()**还有其他方法来指定显示的内容；
> - **Step 4**：调用**setPositive/Negative/NeutralButton()**设置：确定，取消，中立按钮；
> - **Step 5**：调用**create()**方法创建这个对象，再调用**show()**方法将对话框显示出来；

2.几种常用的对话框使用示例

运行效果图：



核心代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_dialog_one;
    private Button btn_dialog_two;
    private Button btn_dialog_three;
    private Button btn_dialog_four;

    private Context mContext;
    private boolean[] checkItems;

    private AlertDialog alert = null;
    private AlertDialog.Builder builder = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        bindView();

    }

    private void bindView() {
        btn_dialog_one = (Button) findViewById(R.id.btn_dialog_one
        btn_dialog_two = (Button) findViewById(R.id.btn_dialog_two
        btn_dialog_three = (Button) findViewById(R.id.btn_dialog_th
        btn_dialog_four = (Button) findViewById(R.id.btn_dialog_fou
```

```
        btn_dialog_one.setOnClickListener(this);
        btn_dialog_two.setOnClickListener(this);
        btn_dialog_three.setOnClickListener(this);
        btn_dialog_four.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            //普通对话框
            case R.id.btn_dialog_one:
                alert = null;
                builder = new AlertDialog.Builder(mContext);
                alert = builder.setIcon(R.mipmap.ic_icon_fish)
                        .setTitle("系统提示：")
                        .setMessage("这是一个最普通的AlertDialog,\n带
                        .setNegativeButton("取消", new DialogInterf
                            @Override
                            public void onClick(DialogInterface dia
                                Toast.makeText(mContext, "你点击了取
                            }
                        })
                        .setPositiveButton("确定", new DialogInterf
                            @Override
                            public void onClick(DialogInterface dia
                                Toast.makeText(mContext, "你点击了确
                            }
                        })
                        .setNeutralButton("中立", new DialogInterfa
                            @Override
                            public void onClick(DialogInterface dia
                                Toast.makeText(mContext, "你点击了中
                            }
                        }).create();             //创建AlertDialog对
                alert.show();                    //显示对话框
                break;
            //普通列表对话框
            case R.id.btn_dialog_two:
                final String[] lesson = new String[]{"语文", "数学",
                alert = null;
                builder = new AlertDialog.Builder(mContext);
                alert = builder.setIcon(R.mipmap.ic_icon_fish)
                        .setTitle("选择你喜欢的课程")
                        .setItems(lesson, new DialogInterface.OnCli
                            @Override
                            public void onClick(DialogInterface dia
                                Toast.makeText(getApplicationContex
                            }
                        }).create();
                alert.show();
                break;
            //单选列表对话框
            case R.id.btn_dialog_three:
```

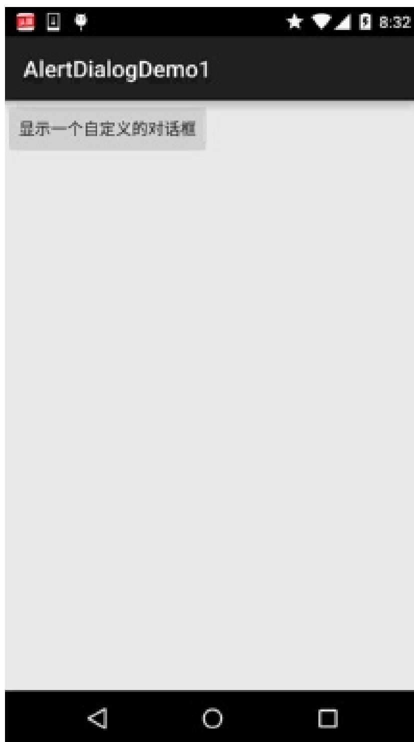
```
                final String[] fruits = new String[]{"苹果", "雪梨",
                alert = null;
                builder = new AlertDialog.Builder(mContext);
                alert = builder.setIcon(R.mipmap.ic_icon_fish)
                        .setTitle("选择你喜欢的水果，只能选一个哦~")
                        .setSingleChoiceItems(fruits, 0, new Dialog
                            @Override
                            public void onClick(DialogInterface dia
                                Toast.makeText(getApplicationContex
                            }
                        }).create();
                alert.show();
                break;
            //多选列表对话框
            case R.id.btn_dialog_four:
                final String[] menu = new String[]{"水煮豆腐", "萝卜
                //定义一个用来记录个列表项状态的boolean数组
                checkItems = new boolean[]{false, false, false, fal
                alert = null;
                builder = new AlertDialog.Builder(mContext);
                alert = builder.setIcon(R.mipmap.ic_icon_fish)
                        .setMultiChoiceItems(menu, checkItems, new
                            @Override
                            public void onClick(DialogInterface dia
                                checkItems[which] = isChecked;
                            }
                        })
                        .setPositiveButton("确定", new DialogInterf
                            @Override
                            public void onClick(DialogInterface dia
                                String result = "";
                                for (int i = 0; i < checkItems.leng
                                    if (checkItems[i])
                                        result += menu[i] + " ";
                                }
                                Toast.makeText(getApplicationContex
                            }
                        })
                        .create();
                alert.show();
                break;
        }
    }
}
```

布局就是四个简单的按钮，这里就不贴出来了，用法非常简单~无非就是创建一个Builder对象后， 进行相关设置，然后create()生成一个AlertDialog对象，最后调用show()方法将AlertDialog 显示出来而已！另外，细心的你可能发现我们点击对话框的外部区域，对话框就会消失，我们 可以为builder设置**setCancelable(false)**即可解决这个问题！

3.通过Builder的setView()定制显示的AlertDialog

> 我们可以自定义一个与系统对话框不同的布局，然后调用setView()将我们的布局加载到 AlertDialog上，上面我们来实现这个效果：

运行效果图：



关键代码：

首先是两种不同按钮的selctor的drawable文件：

btn_selctor_exit.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_pressed="true" android:drawable="@mipmap/iv
    <item android:drawable="@mipmap/iv_icon_exit_normal"/>
</selector>
```

btn_selctor_choose.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_pressed="true" android:drawable="@mipmap/bg
    <item android:drawable="@mipmap/bg_btn_normal"/>
</selector>
```

接着是自定义的Dialog布局：**view_dialog_custom.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <RelativeLayout
        android:id="@+id/titlelayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:background="#53CC66"
        android:padding="5dp">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="提示信息"
            android:textColor="#ffffff"
            android:textSize="18sp"
            android:textStyle="bold" />

        <Button
            android:id="@+id/btn_cancle"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_alignParentRight="true"
            android:background="@drawable/btn_selctor_exit" />

    </RelativeLayout>

    <LinearLayout
        android:id="@+id/ly_detail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/titlelayout"
        android:layout_centerInParent="true"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="20dp"
            android:text="通过setView()方法定制AlertDialog"
            android:textColor="#04AEDA"
```

```
            android:textSize="18sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:text="作者:Coder-pig"
            android:textColor="#04AEDA"
            android:textSize="18sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ly_detail"
        android:layout_marginTop="10dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btn_blog"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:background="@drawable/btn_selctor_choose"
            android:text="访问博客"
            android:textColor="#ffffff"
            android:textSize="20sp" />

        <Button
            android:id="@+id/btn_close"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:background="@drawable/btn_selctor_choose"
            android:text="关闭"
            android:textColor="#ffffff"
            android:textSize="20sp" />

    </LinearLayout>

</RelativeLayout>
```

最后是**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity {

    private Button btn_show;
```

```
    private View view_custom;
    private Context mContext;
    private AlertDialog alert = null;
    private AlertDialog.Builder builder = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        btn_show = (Button) findViewById(R.id.btn_show);

        //初始化Builder
        builder = new AlertDialog.Builder(mContext);

        //加载自定义的那个View,同时设置下
        final LayoutInflater inflater = MainActivity.this.getLayout
        view_custom = inflater.inflate(R.layout.view_dialog_custom,
        builder.setView(view_custom);
        builder.setCancelable(false);
        alert = builder.create();

        view_custom.findViewById(R.id.btn_cancle).setOnClickListene
            @Override
            public void onClick(View v) {
                alert.dismiss();
            }
        });

        view_custom.findViewById(R.id.btn_blog).setOnClickListener(
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "访问博客",
                Uri uri = Uri.parse("http://blog.csdn.net/coder_pig
                Intent intent = new Intent(Intent.ACTION_VIEW, uri)
                startActivity(intent);
                alert.dismiss();
            }
        });

        view_custom.findViewById(R.id.btn_close).setOnClickListener
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "对话框已关闭
                alert.dismiss();
            }
        });

        btn_show.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alert.show();
            }
```

```
            });
        }
    }
```

4.示例代码下载

AlertDialogDemo.zip

AlertDialogDemo1.zip

本节小结：

好的，本节给大家介绍了一下AlertDialog的基本使用，写了几个调用AlertDialog的例子， 最后还通过setView方法自定义了一下我们的AlertDialog！是不是还意犹未尽呢？但这说不上 真正的自定义控件，我们把自定义控件放到进阶系列，到时后会有个专题来和大家探讨 下自定义控件~敬请期待~就说这么多，谢谢~

2.6.0 其他几种常用对话框基本使用

本节引言：

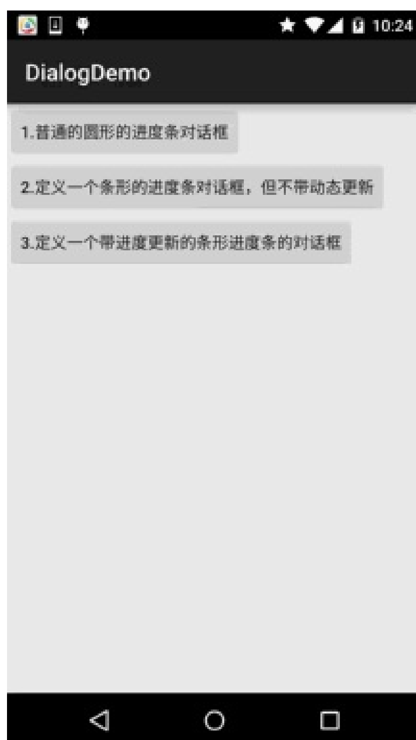
> 上节我们对Dialog的父类：AlertDialog进行了学习，而本节我们来学习下几个常用的 Dialog的基本使用，他们分别是：ProgressDialog(进度条对话框), DatePickerDialog (日期选择对话框)和TimePickerDialog(时间选择对话框)~, 话不多说，开始本节内容~

1.ProgressDialog(进度条对话框)的基本使用

> 我们创建进度条对话框的方式有两种：
>
> - 1.直接调用ProgressDialog提供的静态方法show()显示
> - 2.创建ProgressDialog,再设置对话框的参数,最后show()出来

代码示例：

运行效果图：



关键实现代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements
```

代码比较简单，而关于Progress的东西我们已经在前面学习过了，这里就不啰嗦了~

2.DatePickerDialog(日期选择对话框)与TimePickerDialog(时间选择对话框)

> 先要说明一点： Date/TimePickerDialog只是供用户来选择日期时间,对于android系统的系统时间, 日期没有任何影响,google暂时没有公布系统日期时间设置的API, 如果要在app中设置的话,要重新编译android的系统源码，非常麻烦！
>
> 他们两个的构造方法非常相似： **DatePickerDialog**(上下文；DatePickerDialog.OnDateSetListener()监听器；年；月；日) **TimePickerDialog**(上下文；TimePickerDialog.OnTimeSetListener()监听器；小时，分钟，是否采用24小时制)

代码示例：

运行效果图：



关键实现代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements
```

代码同样很简单，就不解释了~

3.代码下载：

DialogDemo.zip

DialogDemo1.zip

本节小结：

> 好的，本节介绍了三个常用的Dialog，相比起以前的4.x的版本，5.0的这些原生控件，显然要好看得多~就说这么多，谢谢~

2.6.1 PopupWindow(悬浮框)的基本使用

本节引言：

> 本节给大家带来的是最后一个用于显示信息的UI控件——PopupWindow(悬浮框)，如果你想知道 他长什么样子，你可以打开你手机的QQ，长按列表中的某项，这个时候后弹出一个黑色的小 对话框，这种就是PopupWindow了，和AlertDialog对话框不同的是，他的位置可以是随意的；
>
> 另外AlertDialog是非堵塞线程的，而PopupWindow则是堵塞线程的！而官方有这样一句话来介绍 PopupWindow：
>
> **A popup window that can be used to display an arbitrary view. The popup window is**
>
> **a floating container that appears on top of the current activity.**
>
> 大概意思是：一个弹出窗口控件，可以用来显示任意View，而且会浮动在当前activity的顶部
>
> 下面我们就来对这个控件进行学习~
>
> 官方文档：PopupWindow

1.相关方法解读

1）几个常用的构造方法

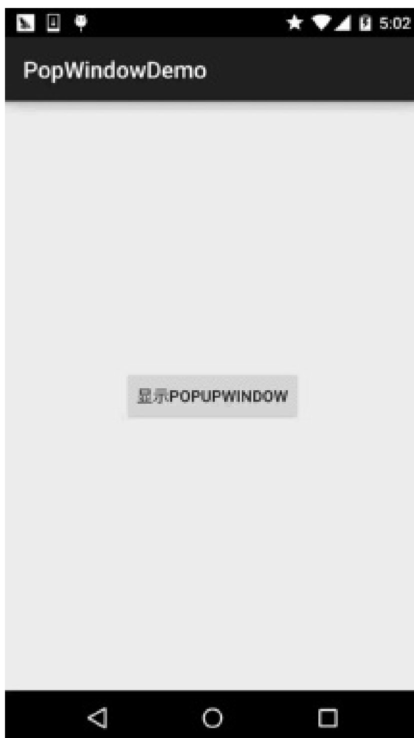
> 我们在文档中可以看到，提供给我们的PopupWindow的构造方法有九种之多，这里只贴实际 开发中用得较多的几个构造方法：
>
> - **public PopupWindow (Context context)**
> - **public PopupWindow(View contentView, int width, int height)**
> - **public PopupWindow(View contentView)**
> - **public PopupWindow(View contentView, int width, int height, boolean focusable)**
>
> 参数就不用多解释了吧，contentView是PopupWindow显示的View，focusable是否显示焦点

2）常用的一些方法

> 下面介绍几个用得较多的一些方法，其他的可自行查阅文档：
>
> - **setContentView**(View contentView)：设置PopupWindow显示的View
> - **getContentView**()：获得PopupWindow显示的View
> - **showAsDropDown(View anchor)**：相对某个控件的位置（正左下方），无偏移
> - **showAsDropDown(View anchor, int xoff, int yoff)**：相对某个控件的位置，有偏移
> - **showAtLocation(View parent, int gravity, int x, int y)**： 相对于父控件的位置（例如正中央Gravity.CENTER，下方Gravity.BOTTOM等），可以设置偏移或无偏移 PS:parent这个参数只要是activity中的view就可以了！
> - **setWidth/setHeight**：设置宽高，也可以在构造方法那里指定好宽高， 除了可以写具体的值，还可以用WRAP_CONTENT或MATCH_PARENT，popupWindow的width和height属性直接和第一层View相对应。
> - **setFocusable(true)**：设置焦点，PopupWindow弹出后，所有的触屏和物理按键都由PopupWindows 处理。其他任何事件的响应都必须发生在PopupWindow消失之后，（home 等系统层面的事件除外）。 比如这样一个PopupWindow出现的时候，按back键首先是让PopupWindow消失，第二次按才是退出 activity，准确的说是想退出activity你得首先让PopupWindow消失，因为不并是任何情况下按back PopupWindow都会消失，必须在PopupWindow设置了背景的情况下 。
> - **setAnimationStyle(int)**：设置动画效果

2.使用代码示例

运行效果图：



实现关键代码：

先贴下动画文件：**anim_pop.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0"
        android:toAlpha="1"
        android:duration="2000">
    </alpha>
</set>
```

接着是popupWindow的布局：**item_popip.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/ic_pop_bg"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_xixi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="嘻嘻"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btn_hehe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:text="呵呵"
        android:textSize="18sp" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private Button btn_show;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        mContext = MainActivity.this;
        btn_show = (Button) findViewById(R.id.btn_show);
        btn_show.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                initPopWindow(v);
            }
        });
    }

    private void initPopWindow(View v) {
        View view = LayoutInflater.from(mContext).inflate(R.layout.
        Button btn_xixi = (Button) view.findViewById(R.id.btn_xixi)
        Button btn_hehe = (Button) view.findViewById(R.id.btn_hehe)
        //1.构造一个PopupWindow，参数依次是加载的View，宽高
        final PopupWindow popWindow = new PopupWindow(view,
                ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.Layo

        popWindow.setAnimationStyle(R.anim.anim_pop);  //设置加载动画

        //这些为了点击非PopupWindow区域，PopupWindow会消失的，如果没有下面
        //代码的话，你会发现，当你把PopupWindow显示出来了，无论你按多少次后
        //PopupWindow并不会关闭，而且退不出程序，加上下述代码可以解决这个问
        popWindow.setTouchable(true);
        popWindow.setTouchInterceptor(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                return false;
                // 这里如果返回true的话，touch事件将被拦截
                // 拦截后 PopupWindow的onTouchEvent不被调用，这样点击外
            }
        });
        popWindow.setBackgroundDrawable(new ColorDrawable(0x0000000

        //设置popupWindow显示的位置，参数依次是参照View，x轴的偏移量，y轴的
        popWindow.showAsDropDown(v, 50, 0);

        //设置popupWindow里的按钮的事件
        btn_xixi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "你点击了嘻嘻~", To
            }
        });
        btn_hehe.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "你点击了呵呵~", To
                popWindow.dismiss();
            }
        });
    }
}
```



3.示例代码下载

PopWindowDemo.zip

本节小结：

> 时间关系，并没有想到好一点的示例，就写了一个简单的demo，应该能满足简单的需要，另外 如果想深入研究下PopupWindow的话可看下述的参考文献：Android PopupWindow的使用和分析 Android PopupWindow详解 嗯，就说这么多，谢谢~

2.6.2 菜单(Menu)

本节引言：

> 本章给大家带来的是Android中的Menu(菜单)，而在Android中的菜单有如下几种：
>
> - **OptionMenu**：选项菜单，android中最常见的菜单，通过Menu键来调用
> - **SubMenu**：子菜单，android中点击子菜单将弹出一个显示子菜单项的悬浮框， 子菜单不支持嵌套，即不能包括其他子菜单
> - **ContextMenu**：上下文菜单，通过长按某个视图组件后出现的菜单，该组件需注册上下文菜单 本节我们来依依学习这几种菜单的用法~ PS：官方文档：menus

1.OptionMenu(选项菜单)

1）如何使用OptionMenu？

> 答：非常简单，重写两个方法就好，其实这两个方法我们在创建项目的时候就会自动生成~ 他们分别是：
>
> - public boolean **onCreateOptionsMenu**(Menu menu)：调用OptionMenu，在这里完成菜单初始化
> - public boolean **onOptionsItemSelected**(MenuItem item)：菜单项被选中时触发，这里完成事件处理
>
> 当然除了上面这两个方法我们可以重写外我们还可以重写这三个方法：
>
> - public void **onOptionsMenuClosed**(Menu menu)：菜单关闭会调用该方法
> - public boolean **onPrepareOptionsMenu**(Menu menu)：选项菜单显示前会调用该方法， 可在这里进行菜单的调整(动态加载菜单列表)
> - public boolean **onMenuOpened**(int featureId, Menu menu)：选项菜单打开以后会调用这个方法

而加载菜单的方式有两种，一种是直接通过编写菜单XML文件，然后调用：**getMenuInflater().inflate(R.menu.menu_main, menu);**加载菜单 或者通过代码动态添加，onCreateOptionsMenu的参数menu，调用add方法添加 菜单，add(菜单项的组号，ID，排序号，标题)，另外如果排序号是按添加顺序排序的话都填0即可！

2）使用示例：

运行效果图：



代码实现：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    //1.定义不同颜色的菜单项的标识:
    final private int RED = 110;
    final private int GREEN = 111;
    final private int BLUE = 112;
    final private int YELLOW = 113;
    final private int GRAY= 114;
    final private int CYAN= 115;
    final private int BLACK= 116;

    private TextView tv_test;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv_test = (TextView) findViewById(R.id.tv_test);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if i
        menu.add(1,RED,4,"红色");
        menu.add(1,GREEN,2,"绿色");
        menu.add(1,BLUE,3,"蓝色");
        menu.add(1,YELLOW,1,"黄色");
```

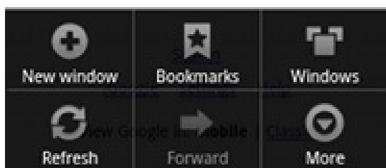
```
        menu.add(1,GRAY,5,"灰色");
        menu.add(1,CYAN,6,"蓝绿色");
        menu.add(1,BLACK,7,"黑色");
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so lo
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        switch (id){
            case RED:
                tv_test.setTextColor(Color.RED);
                break;
            case GREEN:
                tv_test.setTextColor(Color.GREEN);
                break;
            case BLUE:
                tv_test.setTextColor(Color.BLUE);
                break;
            case YELLOW:
                tv_test.setTextColor(Color.YELLOW);
                break;
            case GRAY:
                tv_test.setTextColor(Color.GRAY);
                break;
            case CYAN:
                tv_test.setTextColor(Color.CYAN);
                break;
            case BLACK:
                tv_test.setTextColor(Color.BLACK);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

代码分析：

上述的代码非常简单，给大家演示了Android 5.0的中OptionMenu(选项菜单)中动态添加菜单， 以及事件处理，根据id判断用户点击的是哪一项，然后执行对应的操作！ 另外，有一点要注意的是，选项菜单经过了三个阶段的过渡： 在Android 2.3.x或者更低版本，因为该阶段大部分的机型都是带有Menu键的， 此阶段通过点击Menu键弹出菜单：

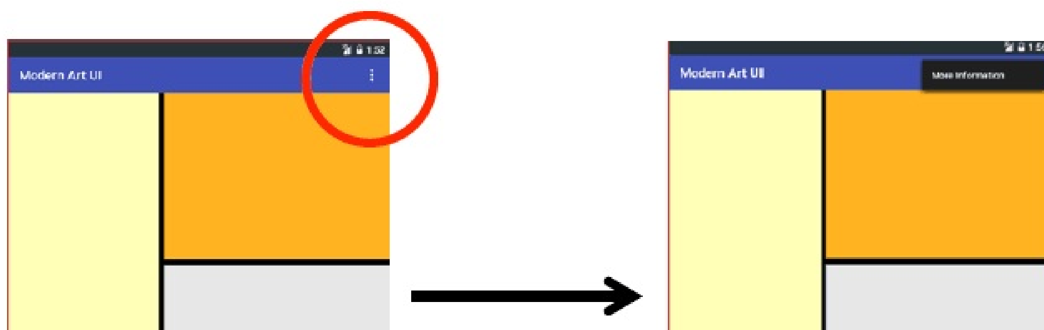




而在Android 3.0或者更高的版本，则是通过3.0引入的ActionBar中的setting菜单：



而在5.0以上的版本则是在ToolBar中的，点击后出一个溢出式的菜单样式



另外通过XML方式定义Menu的方式，我们贴个简单的参考代码：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
          android:icon="@drawable/ic_new_game"
          android:title="@string/new_game"
          android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
          android:icon="@drawable/ic_help"
          android:title="@string/help" />
</menu>
```

其他的自行查阅文档哈~

2.ContextMenu(上下文菜单)

> 一开始我们就说了，长按某个View后出现的菜单，我们需要为这个View注册上下文菜单！

1）如何使用ContextMenu？

> 答：使用的流程如下：
>
> - **Step 1**：重写onCreateContextMenu()方法
> - **Step 2**：为view组件注册上下文菜单，使用registerForContextMenu()方法,参数是View
> - **Step 3**：重写onContextItemSelected()方法为菜单项指定事件监听器

上面的OptionMenu我们使用了Java代码的方法来完成菜单项的添加，这里我们就用XML文件的 方式来生成我们的CotnextMenu吧，另外关于使用Java代码来生成菜单还是使用XML来生成菜单， 建议使用后者来定义菜单，这样可以减少Java代码的代码臃肿，而且不用每次都用代码分配 id，只需修改XML文件即可修改菜单的内容，这样在一定程度上位程序提供的了更好的解耦， 低耦合，高内聚，是吧~

2）使用示例：

运行效果图：



实现代码：

我们先来编写选项菜单的菜单XML文件：

menu_context.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 定义一组单选按钮 -->
    <!-- checkableBehavior的可选值由三个：single设置为单选，all为多选，no
    <group android:checkableBehavior="none">
        <item android:id="@+id/blue" android:title="@string/font_bl
        <item android:id="@+id/green" android:title="@string/font_g
        <item android:id="@+id/red" android:title="@string/font_red
    </group>
</menu>
```

接着我们在选项菜单的那个基础上，添加一个TextView，然后加上下面一些东西：

```
private TextView tv_context;
tv_context = (TextView) findViewById(R.id.tv_context);
        registerForContextMenu(tv_context);

    //重写与ContextMenu相关方法
    @Override
    //重写上下文菜单的创建方法
    public void onCreateContextMenu(ContextMenu menu, View v,
                                    ContextMenu.ContextMenuInfo mer
        MenuInflater inflator = new MenuInflater(this);
        inflator.inflate(R.menu.menu_context, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
    }

    //上下文菜单被点击是触发该方法
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.blue:
                tv_context.setTextColor(Color.BLUE);
                break;
            case R.id.green:
                tv_context.setTextColor(Color.GREEN);
                break;
            case R.id.red:
                tv_context.setTextColor(Color.RED);
                break;
        }
        return true;
    }
```

好的，就是那么简单~可以为多个View设置上下文，switch(v.getId)你懂的~ 另外，和等下要讲的子菜单一样，上下文菜单都无法显示图标！

3.SubMenu(子菜单)

所谓的子菜单只是在**<item>**中又嵌套了一层**<menu>**而已

代码示例：

运行效果图

：

实现代码:

编写子菜单的Menu文件：menu_sub.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/submenu" android:title="子菜单使用演示~">
        <menu>
            <group android:checkableBehavior = "none">
                <item android:id="@+id/one" android:title = "子菜单
                <item android:id="@+id/two" android:title = "子菜单
                <item android:id="@+id/three" android:title = "子菜
            </group>
        </menu>
    </item>
</menu>
```

接着我们改下上面上下文菜单的两个方法的内容，换上下面的代码：

```
public void onCreateContextMenu(ContextMenu menu, View v,
                                    ContextMenu.ContextMenuInfo men
        //子菜单部分：
        MenuInflater inflator = new MenuInflater(this);
        inflator.inflate(R.menu.menu_sub, menu);
        super.onCreateContextMenu(menu, v, menuInfo);
}

public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
            case R.id.one:
                Toast.makeText(MainActivity.this,"你点击了子菜单一",T
                break;
            case R.id.two:
                item.setCheckable(true);
                Toast.makeText(MainActivity.this,"你点击了子菜单二",T
                break;
            case R.id.three:
                Toast.makeText(MainActivity.this,"你点击了子菜单三",T
                item.setCheckable(true);
                break;
        }
    return true;
}
```

好的，灰常简单是吧，另外，如果你想在Java代码中添加子菜单的话，可以调用addSubMenu()

比如：SubMenu file = menu.addSubMenu("文件");file还需要addItem添加菜单项哦！

4.PopupMenu(弹出式菜单)

一个类似于PopupWindow的东东，他可以很方便的在指定View下显示一个弹出菜单，而且 他的菜单选项可以来自于Menu资源，下面我们写个简单的例子来使用下这个东东~

使用示例：

运行效果图：



实现代码：

菜单资源文件：menu_pop.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/lpig" android:title="小猪" />
    <item android:id="@+id/bpig" android:title="大猪" />
</menu>
```

在布局中添加一个按钮，然后添加点击事件：

MainActivity.java:

```
btn_show_menu.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                PopupMenu popup = new PopupMenu(MainActivity.this,b
                popup.getMenuInflater().inflate(R.menu.menu_pop, po
                popup.setOnMenuItemClickListener(new PopupMenu.OnMe
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        switch (item.getItemId()){
                            case R.id.lpig:
                                Toast.makeText(MainActivity.this,"
                                Toast.LENGTH_SHORT).show();
                                break;
                            case R.id.bpig:
                                Toast.makeText(MainActivity.this,"
                                Toast.LENGTH_SHORT).show();
                                break;
                        }
                        return true;
                    }
                });
                popup.show();
            }
        });
```

非常简单，新技能get了没？

5.示例代码下载

MenuDemo1.zip

本节小结：

> 好的，本节给大家介绍了Android中的三种菜单，选项菜单，上下文菜单以及子菜单，最后还讲解了一个PopupMenu的控件，这里只演示了基本的用法，其他属性可自行查阅文档，文档才是最好的老师~嗯，就说这么多，谢谢，对了今天国庆，祝大家国庆玩得开心！

2.6.3 ViewPager的简单使用

本节引言：

> 本节带来的是Android 3.0后引入的一个UI控件——ViewPager(视图滑动切换工具)，实在想不到 如何来称呼这个控件，他的大概功能：通过手势滑动可以完成View的切换，一般是用来做APP 的引导页或者实现图片轮播，因为是3.0后引入的，如果想在低版本下使用，就需要引入v4 兼容包哦~，我们也可以看到，ViewPager在：android.support.v4.view.ViewPager目录下~ 下面我们就来学习一下这个控件的基本用法~ 官方API文档：ViewPager

1.ViewPager的简单介绍

> ViewPager就是一个简单的页面切换组件，我们可以往里面填充多个View，然后我们可以左 右滑动，从而切换不同的View，我们可以通过setPageTransformer()方法为我们的ViewPager 设置切换时的动画效果，当然，动画我们还没学到，所以我们把为ViewPager设置动画 放到下一章绘图与动画来讲解！和前面学的ListView，GridView一样，我们也需要一个Adapter (适配器)将我们的View和ViewPager进行绑定，而ViewPager则有一个特定的Adapter—— **PagerAdapter**！另外，Google官方是建议我们使用Fragment来填充ViewPager的，这样 可以更加方便的生成每个Page，以及管理每个Page的生命周期！给我们提供了两个Fragment 专用的Adapter：**FragmentPageAdapter和FragmentStatePagerAdapter** 我们简要的来分析下这两个Adapter的区别：
>
> - **FragmentPageAdapter**：和PagerAdapter一样，只会缓存当前的Fragment以及左边一个，右边 一个，即总共会缓存3个Fragment而已，假如有1，2，3，4四个页面： 处于1页面：缓存1，2 处于2页面：缓存1，2，3 处于3页面：销毁1页面，缓存2，3，4 处于4页面：销毁2页面，缓存3，4 更多页面的情况，依次类推~
> - **FragmentStatePagerAdapter**：当Fragment对用户不 见得时，整个Fragment会被销毁， 只会保存Fragment的状态！而在页面需要重新显示的时候，会生成新的页面！
>
> 综上，FragmentPageAdapter适合固定的页面较少的场合；而FragmentStatePagerAdapter则适合 于页面较多或者页面内容非常复杂(需占用大量内存)的情况！

2.PagerAdapter的使用

> 我们先来介绍最普通的PagerAdapter，如果想使用这个PagerAdapter需要重写下面的四个方法：当然，这只是官方建议，实际上我们只需重写getCount()和isViewFromObject()就可以了~
>
> - **getCount()**:获得viewpager中有多少个view
> - **destroyItem()**:移除一个给定位置的页面。适配器有责任从容器中删除这个视图。这是为了确保在finishUpdate(viewGroup)返回时视图能够被移除。
>
> 而另外两个方法则是涉及到一个key的东东：
>
> - **instantiateItem()**: ①将给定位置的view添加到ViewGroup(容器)中,创建并显示出来 ②返回一个代表新增页面的Object(key),通常都是直接返回view本身就可以了,当然你也可以 自定义自己的key,但是key和每个view要一一对应的关系
> - **isViewFromObject()**: 判断instantiateItem(ViewGroup, int)函数所返回来的Key与一个页面视图是否是 代表的同一个视图(即它俩是否是对应的，对应的表示同一个View),通常我们直接写 return view == object!

使用示例1：最简单用法

运行效果图：



关键部分代码：

好的，代码写起来也是非常简单的：首先是每个View的布局，一式三份，另外两个View一样：

view_one.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFBA55"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="第一个Page"
        android:textColor="#000000"
        android:textSize="18sp"
        android:textStyle="bold" />
</LinearLayout>
```

然后编写一个自定义个的PagerAdapter：

MyPagerAdapter.java：

```
public class MyPagerAdapter extends PagerAdapter {
    private ArrayList<View> viewLists;

    public MyPagerAdapter() {
    }

    public MyPagerAdapter(ArrayList<View> viewLists) {
        super();
        this.viewLists = viewLists;
    }

    @Override
    public int getCount() {
        return viewLists.size();
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }

    @Override
    public Object instantiateItem(ViewGroup container, int positior
        container.addView(viewLists.get(position));
        return viewLists.get(position);
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Obje
        container.removeView(viewLists.get(position));
    }
}
```

接着到Activity了，和以前学的ListView非常类似：

OneActivity.java：

```
public class OneActivity extends AppCompatActivity{

    private ViewPager vpager_one;
    private ArrayList<View> aList;
    private MyPagerAdapter mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
        vpager_one = (ViewPager) findViewById(R.id.vpager_one);

        aList = new ArrayList<View>();
        LayoutInflater li = getLayoutInflater();
        aList.add(li.inflate(R.layout.view_one,null,false));
        aList.add(li.inflate(R.layout.view_two,null,false));
        aList.add(li.inflate(R.layout.view_three,null,false));
        mAdapter = new MyPagerAdapter(aList);
        vpager_one.setAdapter(mAdapter);
    }
}
```

好的，关键代码就上述部分，非常容易理解~

使用示例2：标题栏——PagerTitleStrip与PagerTabStrip

> 就是跟随着ViewPager滑动而滑动的标题咯，这两个是官方提供的，一个是普通文字， 一个是带有下划线，以及可以点击文字可切换页面，下面我们来写个简单的例子~

运行效果图：



关键代码实现：

这里两者的区别仅仅是布局不一样而已，其他的都一样：

PagerTitleStrip所在Activtiy的布局：**activity_two.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="#CCFF99"
        android:gravity="center"
        android:text="PagerTitleStrip效果演示"
        android:textColor="#000000"
        android:textSize="18sp" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager_two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <android.support.v4.view.PagerTitleStrip
            android:id="@+id/pagertitle"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:layout_gravity="top"
            android:textColor="#FFFFFF" />
   </android.support.v4.view.ViewPager>

</LinearLayout>
```

而PagerTabStrip所在的布局：

activity_three.xml：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="35dp"
        android:background="#C0C080"
        android:gravity="center"
        android:text="PagerTabStrip效果演示"
        android:textSize="18sp" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager_three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <android.support.v4.view.PagerTabStrip
            android:id="@+id/pagertitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="top" />
     </android.support.v4.view.ViewPager>
</LinearLayout>
```

接下来的两者都一样了，我们先来编写一个自定义的PagerAdapter，除了前面重写的 四个方法外，我们需要另外重写一个方法：**getPageTitle**()，这个设置标题的~代码如下：

MyPagerAdapter2.java：

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class MyPagerAdapter2 extends PagerAdapter {
    private ArrayList<View> viewLists;
    private ArrayList<String> titleLists;

    public MyPagerAdapter2() {}
    public MyPagerAdapter2(ArrayList<View> viewLists,ArrayList<Stri
    {
        this.viewLists = viewLists;
        this.titleLists = titleLists;
    }

    @Override
    public int getCount() {
        return viewLists.size();
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }

    @Override
    public Object instantiateItem(ViewGroup container, int positior
        container.addView(viewLists.get(position));
        return viewLists.get(position);
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Obje
        container.removeView(viewLists.get(position));
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return titleLists.get(position);
    }
}
```

最后是Activity部分，两个都是一样的：

TwoActivity.java：

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class TwoActivity extends AppCompatActivity {

    private ViewPager vpager_two;
    private ArrayList<View> aList;
    private ArrayList<String> sList;
    private MyPagerAdapter2 mAdapter;

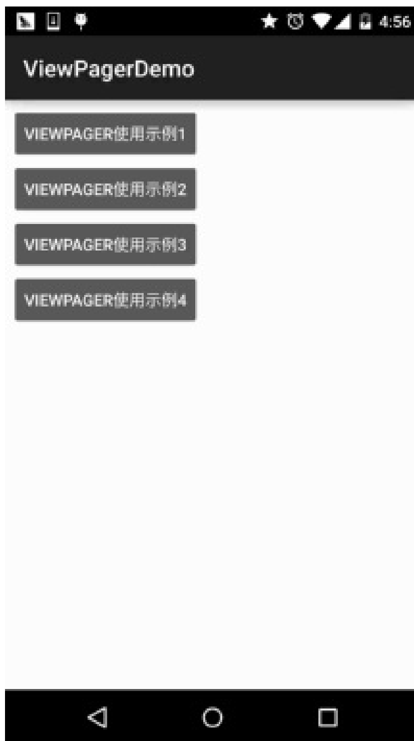
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);
        vpager_two = (ViewPager) findViewById(R.id.vpager_two);
        aList = new ArrayList<View>();
        LayoutInflater li = getLayoutInflater();
        aList.add(li.inflate(R.layout.view_one,null,false));
        aList.add(li.inflate(R.layout.view_two,null,false));
        aList.add(li.inflate(R.layout.view_three, null, false));
        sList = new ArrayList<String>();
        sList.add("橘黄");
        sList.add("淡黄");
        sList.add("浅棕");
        mAdapter = new MyPagerAdapter2(aList,sList);
        vpager_two.setAdapter(mAdapter);
    }
}
```

好了，非常简单，有疑问的话，自己下demo看看就懂了~

使用示例3：ViewPager实现TabHost的效果：

> 当然，示例2很多时候，只是中看不中用，实际开发中我们可能需要自行定制这个标题栏， 下面我们就来写个简单的例子来实现TabHost的效果，如果你不知道TabHost是什么鬼的 话，那么，请看效果图！

运行效果图：



实现逻辑解析：

下面我们来讲解下实现上述效果的逻辑，然后贴代码：

> 首先是布局：顶部一个LinearLayout，包着三个TextView，weight属性都为1，然后下面跟着 一个滑块的ImageView，我们设置宽度为match_parent；最底下是我们的ViewPager，这里可能 有两个属性你并不认识，一个是：flipInterval：这个是指定View动画间的时间间隔的！ 而persistentDrawingCache：则是设置控件的绘制缓存策略，可选值有四个：
>
> - none：不在内存中保存绘图缓存；
> - animation:只保存动画绘图缓存；
> - scrolling：只保存滚动效果绘图缓存；
> - all：所有的绘图缓存都应该保存在内存中；
>
> 可以同时用2个，animation|scrolling这样~

布局代码：**activity_four.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="48dp"
        android:background="#FFFFFF">
```

```
        <TextView
            android:id="@+id/tv_one"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="橘黄"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/tv_two"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="淡黄"
            android:textColor="#000000" />

        <TextView
            android:id="@+id/tv_three"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1.0"
            android:gravity="center"
            android:text="浅棕"
            android:textColor="#000000" />
    </LinearLayout>

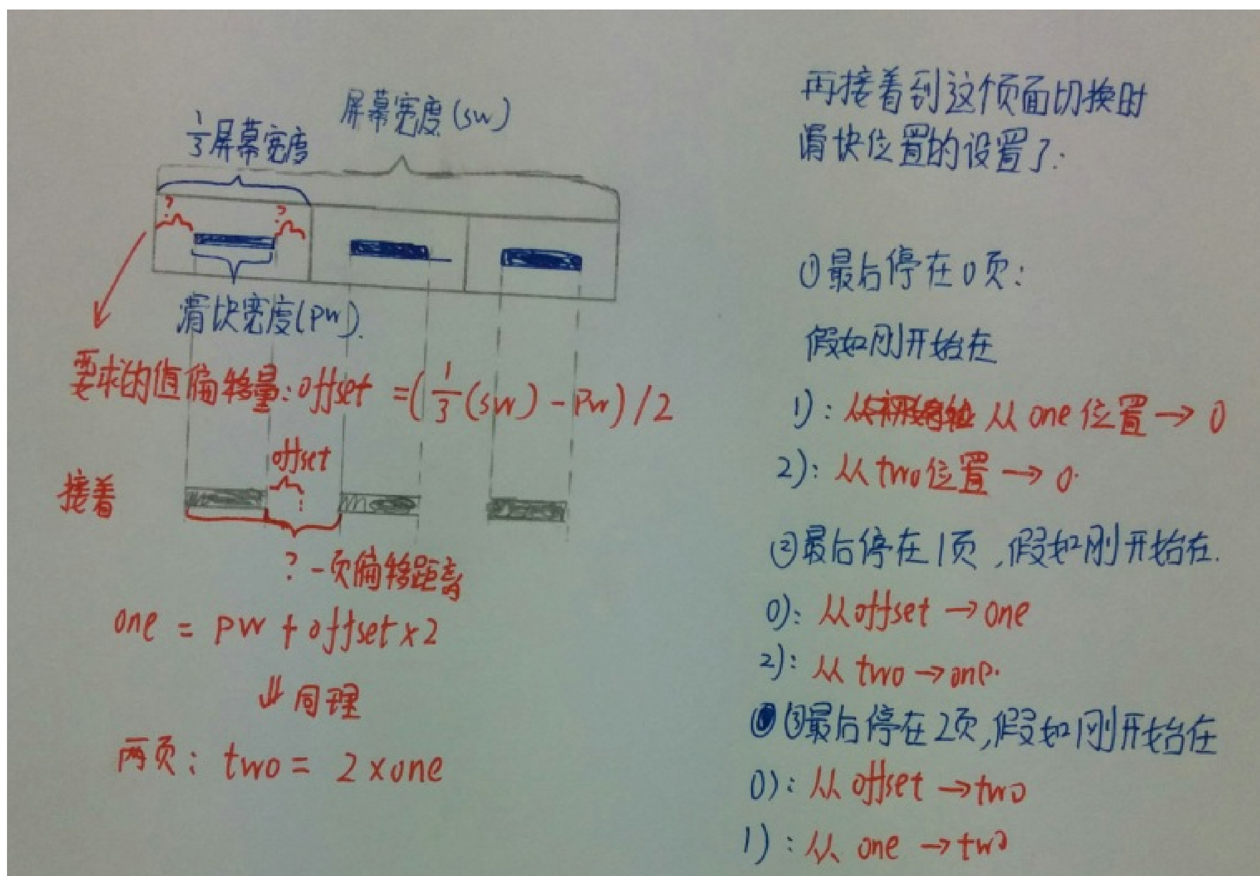
    <ImageView
        android:id="@+id/img_cursor"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:scaleType="matrix"
        android:src="@mipmap/line" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager_four"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_gravity="center"
        android:layout_weight="1.0"
        android:flipInterval="30"
        android:persistentDrawingCache="animation" />

</LinearLayout>
```

> 接着到我们的Activity了，我们来捋下思路：
>
> - **Step 1**：我们需要让我们的移动块在第一个文字下居中，那这里就要算一下偏移量：先获得图片宽度pw，然后获取屏幕宽度sw，计算方法很简单：**offset(偏移量) = ((sw / 3)-pw) / 2** //屏幕宽/3 - 图片宽度，然后再除以2，左右嘛！然后我么你调用setImageMatrix设置滑块当前的位置：同时我们也把切换一页和两页，滑块的移动距离也算出来，很简单：**one = offset * 2 + pw; two = one * 2;**
> - **Step 2**：当我们滑动页面时，我们的滑块要进行移动，我们要为ViewPager添加一个 OnPageChangeListener事件，我们需要对滑动后的页面来做一个判断，同时记录滑动前处于 哪个页面，下面自己画了个图，可能更容易理解吧！

PS:太久没写字，字很丑，能看清就好，字丑人美，哈哈~ 



嗯，如果还是不能理解的话，自己动手画画图就知道了，下面上代码：

FourActvitiy.java：

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class FourActivity extends AppCompatActivity implements View
```

```
        ViewPager.OnPageChangeListener {

    private ViewPager vpager_four;
    private ImageView img_cursor;
    private TextView tv_one;
    private TextView tv_two;
    private TextView tv_three;

    private ArrayList<View> listViews;
    private int offset = 0;//移动条图片的偏移量
    private int currIndex = 0;//当前页面的编号
    private int bmpWidth;// 移动条图片的长度
    private int one = 0; //移动条滑动一页的距离
    private int two = 0; //滑动条移动两页的距离

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_four);
        initViews();
    }

    private void initViews() {
        vpager_four = (ViewPager) findViewById(R.id.vpager_four);
        tv_one = (TextView) findViewById(R.id.tv_one);
        tv_two = (TextView) findViewById(R.id.tv_two);
        tv_three = (TextView) findViewById(R.id.tv_three);
        img_cursor = (ImageView) findViewById(R.id.img_cursor);

        //下划线动画的相关设置：
        bmpWidth = BitmapFactory.decodeResource(getResources(), R.r
        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);
        int screenW = dm.widthPixels;// 获取分辨率宽度
        offset = (screenW / 3 - bmpWidth) / 2;// 计算偏移量
        Matrix matrix = new Matrix();
        matrix.postTranslate(offset, 0);
        img_cursor.setImageMatrix(matrix);// 设置动画初始位置
        //移动的距离
        one = offset * 2 + bmpWidth;// 移动一页的偏移量,比如1->2,或者2-
        two = one * 2;// 移动两页的偏移量,比如1直接跳3

        //往ViewPager填充View，同时设置点击事件与页面切换事件
        listViews = new ArrayList<View>();
        LayoutInflater mInflater = getLayoutInflater();
        listViews.add(mInflater.inflate(R.layout.view_one, null, fa
        listViews.add(mInflater.inflate(R.layout.view_two, null, fa
        listViews.add(mInflater.inflate(R.layout.view_three, null,
        vpager_four.setAdapter(new MyPagerAdapter(listViews));
        vpager_four.setCurrentItem(0);          //设置ViewPager当前页

        tv_one.setOnClickListener(this);
        tv_two.setOnClickListener(this);
```

```
        tv_three.setOnClickListener(this);

        vpager_four.addOnPageChangeListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.tv_one:
                vpager_four.setCurrentItem(0);
                break;
            case R.id.tv_two:
                vpager_four.setCurrentItem(1);
                break;
            case R.id.tv_three:
                vpager_four.setCurrentItem(2);
                break;
        }
    }

    @Override
    public void onPageSelected(int index) {
        Animation animation = null;
        switch (index) {
            case 0:
                if (currIndex == 1) {
                    animation = new TranslateAnimation(one, 0, 0, (
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, 0, 0, (
                }
                break;
            case 1:
                if (currIndex == 0) {
                    animation = new TranslateAnimation(offset, one,
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, one, 0,
                }
                break;
            case 2:
                if (currIndex == 0) {
                    animation = new TranslateAnimation(offset, two,
                } else if (currIndex == 1) {
                    animation = new TranslateAnimation(one, two, 0,
                }
                break;
        }
        currIndex = index;
        animation.setFillAfter(true);// true表示图片停在动画结束位置
        animation.setDuration(300); //设置动画时间为300毫秒
        img_cursor.startAnimation(animation);//开始动画
    }

    @Override
```

```
    public void onPageScrollStateChanged(int i) {

    }

    @Override
    public void onPageScrolled(int i, float v, int i1) {

    }
}
```

嗯，关于动画可能你并不熟悉，没事，下一章我们带大家一起扣~

3.ViewPager结合Fragment示例

> 嗯，在前面讲解Fragment的时候我们就讲解了一个使用示例：Android基础入门教程——5.2.4 Fragment实例精讲——底部导航栏+ViewPager滑动切换页面 这里就不再详述了，有兴趣的点下链接看看即可~

4.代码示例下载

ViewPagerDemo.zip

本节小结：

> 关于ViewPager，限于篇幅，有些地方并没有讲到，其他的大家需要自己查阅文档了~
>
> 另外，上面也说了，ViewPager的动画我们会在下一章讲解！好的，就说这么多~
>
> 嗯，国庆前曾说会在国庆假期里完成整个系列，结果一篇都没写，实在抱歉...
>
> 因为妹子过来玩了，So，你懂的~，会加快进度~，争取早日进阶！

2.6.4 DrawerLayout(官方侧滑菜单)的简单使用

本节引言：

> 本节给大家带来基础UI控件部分的最后一个控件：**DrawerLayout**，官方给我们提供的一个侧滑菜单 控件，和上一节的ViewPager一样，3.0以后引入，低版本使用它，需要v4兼容包，说到侧滑，相信 很多人都用过github上的SlidingMenu，不过好像有两个版本，一个是单独的，另一个需要依赖另一 个开源项目：ActionBarSherlock；既然Google为我们提供了这个控件，为何不用咧，而且在 Material Design设计规范中，随处可见的很多侧滑菜单的动画效果，大都可以通过Toolbar + DrawerLayout来实现~，本节我们就来探究下这个DrawerLayout的一个基本用法~还有人喜欢把他 称为抽屉控件~官方文档：DrawerLayout

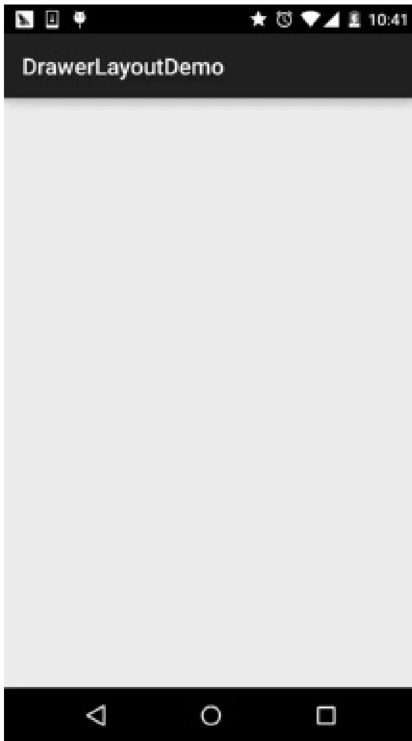
1.使用的注意事项

> - **1**.主内容视图一定要是DrawerLayout的第一个子视图
> - **2**.主内容视图宽度和高度需要match_parent
> - **3**.必须显示指定侧滑视图的android:**layout_gravity**属性 android:layout_gravity = "start"时，从左向右滑出菜单 android:layout_gravity = "end"时，从右向左滑出菜单 不推荐使用left和right!!!
> - 侧滑视图的宽度以dp为单位，不建议超过**320dp**(为了总能看到一些主内容视图)
> - 设置侧滑事件：mDrawerLayout.setDrawerListener(DrawerLayout.DrawerListener);
> - 要说一点：可以结合Actionbar使用当用户点击Actionbar上的应用图标，弹出侧滑菜单！ 这里就要通过**ActionBarDrawerToggle**，它是DrawerLayout.DrawerListener的具体实现类， 我们可以重写ActionBarDrawerToggle的onDrawerOpened()和onDrawerClosed()以监听抽屉拉出 或隐藏事件！但是这里我们不讲，因为5.0后我们使用的是Toolbar！有兴趣的可以自行查阅相关 文档！

2.使用代码示例

示例1：单个侧滑菜单的实现

运行效果图：



实现关键代码：

首先是我们的主布局，注意：最外层要是DrawerLayout哦！！！！

activity_main.xml：

```
<android.support.v4.widget.DrawerLayout xmlns:android="http://scher
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/ly_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <ListView
        android:id="@+id/list_left_drawer"
        android:layout_width="180dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="#080808"
        android:choiceMode="singleChoice"
        android:divider="#FFFFFF"
        android:dividerHeight="1dp" />

</android.support.v4.widget.DrawerLayout>
```

接着ListView的布局代码和domain类：Item比较简单，就不给出了，直接上中间Fragment的 布局以及代码吧！另外Adapter直接复用我们之前写的那个可复用的MyAdapter！

fg_content.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="25sp" />

</RelativeLayout>
```

ContentFragment.java：

```
/**
 * Created by Jay on 2015/10/8 0008.
 */
public class ContentFragment extends Fragment {

    private TextView tv_content;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_content, containe
        tv_content = (TextView) view.findViewById(R.id.tv_content)
        String text = getArguments().getString("text");
        tv_content.setText(text);
        return view;
    }
}
```

最后是我们的Activity类

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Adap

    private DrawerLayout drawer_layout;
    private ListView list_left_drawer;
    private ArrayList<Item> menuLists;
    private MyAdapter<Item> myAdapter = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        drawer_layout = (DrawerLayout) findViewById(R.id.drawer_lay
        list_left_drawer = (ListView) findViewById(R.id.list_left_d

        menuLists = new ArrayList<Item>();
        menuLists.add(new Item(R.mipmap.iv_menu_realtime,"实时信息")
        menuLists.add(new Item(R.mipmap.iv_menu_alert,"提醒通知"));
        menuLists.add(new Item(R.mipmap.iv_menu_trace,"活动路线"));
        menuLists.add(new Item(R.mipmap.iv_menu_settings,"相关设置")
        myAdapter = new MyAdapter<Item>(menuLists,R.layout.item_lis
            @Override
            public void bindView(ViewHolder holder, Item obj) {
                holder.setImageResource(R.id.img_icon,obj.getIconIc
                holder.setText(R.id.txt_content, obj.getIconName()
            }
        };
        list_left_drawer.setAdapter(myAdapter);
        list_left_drawer.setOnItemClickListener(this);
    }

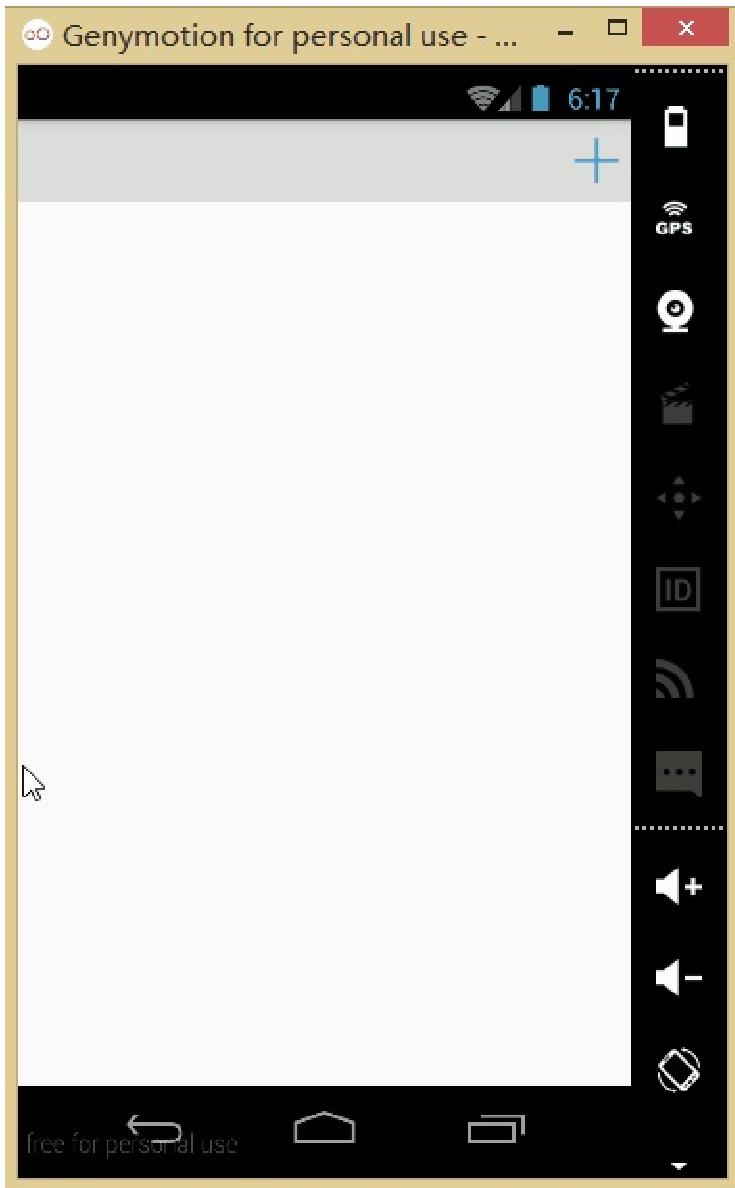
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int p
        ContentFragment contentFragment = new ContentFragment();
        Bundle args = new Bundle();
        args.putString("text", menuLists.get(position).getIconName(
        contentFragment.setArguments(args);
        FragmentManager fm = getSupportFragmentManager();
        fm.beginTransaction().replace(R.id.ly_content,contentFragme
        drawer_layout.closeDrawer(list_left_drawer);
    }
}
```

代码很简单，就不多说了~

示例2.左右两个侧滑菜单的实现

> 嗯，不知道你有没有发现，从上面的DrawerLayout的布局，我们大概可以猜到，DrawerLayout 最多由三个部分组成，中间的内容部分，左边的侧滑菜单部分，右边的侧滑菜单部分组成！下面我们来写一个带有两个侧滑菜单的示例！

运行效果图：



代码实现：

首先我们创建两个Fragment以及对应的布局，他们分别是左右侧滑菜单！

左边**Fragment**：

布局：**fg_left.xml**，这里就用了一个图片而以，点击后弹出一个新的Activity；当然你可以根据自己的需求进行扩展！

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:orientation="vertical" android:layout_width="match_pare
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/img_bg"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@mipmap/bg_menu_left"/>

</LinearLayout>
```

对应的**LeftFragment.java**：

```
/**
 * Created by Jay on 2015/10/9 0009.
 */
public class LeftFragment extends Fragment{

    private DrawerLayout drawer_layout;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup con
        View view = inflater.inflate(R.layout.fg_left, container, f
        ImageView img_bg = (ImageView) view.findViewById(R.id.img_b
        img_bg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getActivity().startActivity(new Intent(getActivity(
                drawer_layout.closeDrawer(Gravity.START);
            }
        });
        return view;
    }

    //暴露给Activity，用于传入DrawerLayout，因为点击后想关掉DrawerLayout
    public void setDrawerLayout(DrawerLayout drawer_layout){
        this.drawer_layout = drawer_layout;
    }
}
```

右面的**Fragment**：

布局就三个按钮，点击后替换中间部分的Fragment，布局**fg_right.xml**代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2F9AF2"
    android:gravity="center"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_one"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单项一" />

    <Button
        android:id="@+id/btn_two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单项二" />

    <Button
        android:id="@+id/btn_three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="菜单项三" />

</LinearLayout>
```

然后对应的是**RightFragment.java**：

```
/**
 * Created by Jay on 2015/10/9 0009.
 */
public class RightFragment extends Fragment implements View.OnClick

    private DrawerLayout drawer_layout;
    private FragmentManager fManager;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_right, container,
        view.findViewById(R.id.btn_one).setOnClickListener(this);
        view.findViewById(R.id.btn_two).setOnClickListener(this);
        view.findViewById(R.id.btn_three).setOnClickListener(this);
        fManager = getActivity().getSupportFragmentManager();
        return view;
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_one:
                ContentFragment cFragment1 = new ContentFragment("
                fManager.beginTransaction().replace(R.id.fly_conter
                drawer_layout.closeDrawer(Gravity.END);
                break;
            case R.id.btn_two:
                ContentFragment cFragment2 = new ContentFragment("
                fManager.beginTransaction().replace(R.id.fly_conter
                drawer_layout.closeDrawer(Gravity.END);
                break;
            case R.id.btn_three:
                ContentFragment cFragment3 = new ContentFragment("
                fManager.beginTransaction().replace(R.id.fly_conter
                drawer_layout.closeDrawer(Gravity.END);
                break;
        }
    }

    public void setDrawerLayout(DrawerLayout drawer_layout){
        this.drawer_layout = drawer_layout;
    }

}
```

另外还有一个中间部分填充的ContentFragment，布局：**fg_content.xml**如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="25sp" />

</RelativeLayout>
```

ContentFragment.java：

```
public class ContentFragment extends Fragment {

    private TextView tv_content;
    private String strContent;
    private int bgColor;

    public ContentFragment(String strContent,int bgColor) {
        this.strContent = strContent;
        this.bgColor = bgColor;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup con
        View view = inflater.inflate(R.layout.fg_content, containe
        view.setBackgroundColor(getResources().getColor(bgColor));
        tv_content = (TextView) view.findViewById(R.id.tv_content);
        tv_content.setText(strContent);
        return view;
    }
}
```

编写好以后，就到我们的Activity的布局了以及Activity的代码了： 在此之前我们还需要些一个顶部条形栏的布局：

view_topbar.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#DCDEDB">

    <Button
        android:id="@+id/btn_right"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:background="@drawable/btn_selctor"/>

</RelativeLayout>
```

然后是**activity_main.xml**：

```
<android.support.v4.widget.DrawerLayout xmlns:android="http://scher
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <include
            android:id="@+id/topbar"
            layout="@layout/view_topbar"
            android:layout_width="wrap_content"
            android:layout_height="48dp" />

        <FrameLayout
            android:id="@+id/fly_content"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>

    <fragment
        android:id="@+id/fg_left_menu"
        android:name="jay.com.drawerlayoutdemo2.LeftFragment"
        android:layout_width="300dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:tag="LEFT"
        tools:layout="@layout/fg_left" />

    <fragment
        android:id="@+id/fg_right_menu"
        android:name="jay.com.drawerlayoutdemo2.RightFragment"
        android:layout_width="100dp"
        android:layout_height="match_parent"
        android:layout_gravity="end"
        android:tag="RIGHT"
        tools:layout="@layout/fg_right" />

</android.support.v4.widget.DrawerLayout>
```

最后是**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity implements Viev

    private DrawerLayout drawer_layout;
    private FrameLayout fly_content;
```

```
    private View topbar;
    private Button btn_right;
    private RightFragment fg_right_menu;
    private LeftFragment fg_left_menu;
    private FragmentManager fManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fManager = getSupportFragmentManager();
        fg_right_menu = (RightFragment) fManager.findFragmentById(R
        fg_left_menu = (LeftFragment) fManager.findFragmentById(R.i
        initViews();
    }

    private void initViews() {
        drawer_layout = (DrawerLayout) findViewById(R.id.drawer_lay
        fly_content = (FrameLayout) findViewById(R.id.fly_content);
        topbar = findViewById(R.id.topbar);
        btn_right = (Button) topbar.findViewById(R.id.btn_right);
        btn_right.setOnClickListener(this);

        //设置右面的侧滑菜单只能通过编程来打开
        drawer_layout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCK
                Gravity.END);

        drawer_layout.setDrawerListener(new DrawerLayout.DrawerList
            @Override
            public void onDrawerSlide(View view, float v) {

            }

            @Override
            public void onDrawerOpened(View view) {

            }

            @Override
            public void onDrawerClosed(View view) {
                drawer_layout.setDrawerLockMode(
                        DrawerLayout.LOCK_MODE_LOCKED_CLOSED, Gravi
            }

            @Override
            public void onDrawerStateChanged(int i) {

            }
        });

        fg_right_menu.setDrawerLayout(drawer_layout);
        fg_left_menu.setDrawerLayout(drawer_layout);
    }
```

```
    @Override
    public void onClick(View v) {
        drawer_layout.openDrawer(Gravity.RIGHT);
        drawer_layout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNL(
                Gravity.END);    //解除锁定
    }
}
```

好的，至此就大功告成了~，呼呼，下面说下看代码时可能会有的疑惑：

> - **1**. drawer_layout.**openDrawer**(Gravity.END); 这句是设置打开的哪个菜单START代表左边，END代表右边
> - **2**. drawer_layout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CLOSED,Gravity.END); 锁定右面的侧滑菜单，不能通过手势关闭或者打开，只能通过代码打开！即调用openDrawer方法！ 接着 drawer_layout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLOCKED,Gravity.END); 解除锁定状态，即可以通过手势关闭侧滑菜单 最后在drawer关闭的时候调用：drawer_layout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CLOSED, Gravity.END); 再次锁定右边的侧滑菜单！
> - **3**. 布局代码中的Tag属性的作用？ 答：这里没用到，在重写DrawerListener的onDrawerSlide方法时，我们可以通过他的第一个 参数drawerView，调用drawerView.getTag().equals("START")判断触发菜单事件的是哪个 菜单！然后可以进行对应的操作！

3.代码示例下载

DrawerLayoutDemo.zip

DrawerLayoutDemo2.zip

本节小结：

> 好的，本节给大家介绍了官方的侧滑控件DrawerLayout的基本用法，使用起来非常的方便！ 当然这里仅仅是简单的使用演示， 另外看到弘扬大神写过一篇：Android DrawerLayout 高仿QQ5.2双向侧滑菜单 有兴趣可以看看，如果看完本节的内容，相信你看起来不会怎么吃力~好的！
>
> 本节就到这里，跟UI控件这一章说拜拜了~下一章我们开始绘图与动画了， 为我们进阶部分的自定义控件系列打基础！ 

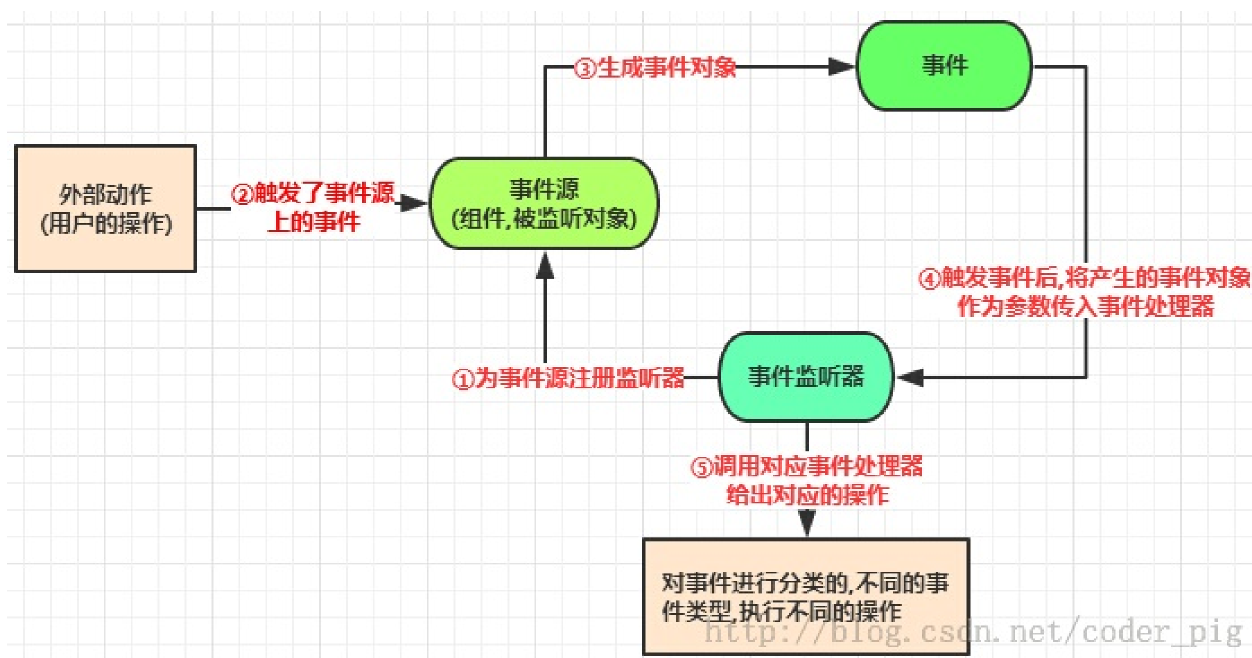
3.1.1 基于监听的事件处理机制

本节引言：

> 第二章我们学习的是Android的UI控件，我们可以利用这些控件构成一个精美的界面，但是仅仅是界面而已；下一步就要开始学习逻辑与业务实现了，本章节讲解的是Android的事件处理机制！何为事件处理机制？举个 简单的例子，比如点击一个按钮，我们向服务器发送登陆请求！当然，Android中的事件处理机制不止这一种， 比如屏幕发生选择，我们点击了屏幕上某个区域...简单点说，事件处理机制就是我们和UI发生交互时，我们在背后添加一些小动作而已！本节我们来介绍使用的最频繁的一种：基于监听的事件处理机制！

1.基于监听的时间处理机制模型:

流程模型图：



文字表述：

> 事件监听机制中由事件源，事件，事件监听器三类对象组成 处理流程如下: **Step 1:**为某个事件源(组件)设置一个监听器,用于监听用户操作 **Step 2:**用户的操作,触发了事件源的监听器 **Step 3:**生成了对应的事件对象 **Step 4:**将这个事件源对象作为参数传给事件监听器 **step 5:**事件监听器对事件对象进行判断,执行对应的事件处理器(对应事件的处理方法)

归纳：

> 事件监听机制是一种委派式的事件处理机制,事件源(组件)事件处理委托给事件监听器 当事件源发生指定事件时,就通知指定事件监听器,执行相应的操作

2.五种不同的使用形式：

我们以下面这个： 简单的按钮点击,提示**Toast**信息的程序；使用五种不同的形式来实现！

效果图：



1）直接用匿名内部类

> 平时最常用的一种:直接setXxxListener后,重写里面的方法即可; 通常是临时使用一次,复用性不高！

实现代码如下：**MainAcivity.java:**

```
package com.jay.example.innerlisten;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
import android.app.Activity;

public class MainActivity extends Activity {
    private Button btnshow;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnshow = (Button) findViewById(R.id.btnshow);
        btnshow.setOnClickListener(new OnClickListener() {
            //重写点击事件的处理方法onClick()
            @Override
            public void onClick(View v) {
                //显示Toast信息
                Toast.makeText(getApplicationContext(), "你点击了按钮
            }
        });
    }
}
```

2）使用内部类

> 和上面的匿名内部类不同哦！ 使用优点:可以在该类中进行复用,可直接访问外部类的所有界面组件！

实现代码如下：**MainAcivity.java:**

```
package com.jay.example.innerlisten;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
import android.app.Activity;

public class MainActivity extends Activity {
    private Button btnshow;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnshow = (Button) findViewById(R.id.btnshow);
        //直接new一个内部类对象作为参数
        btnshow.setOnClickListener(new BtnClickListener());
    }
    //定义一个内部类,实现View.OnClickListener接口,并重写onClick()方法
    class BtnClickListener implements View.OnClickListener
    {
        @Override
        public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "按钮被点击了", T
        }
    }
}
```

3）使用外部类：

> 就是另外创建一个处理事件的Java文件,这种形式用的比较少！因为外部类不能直接访问用户界面 类中的组件,要通过构造方法将组件传入使用;这样导致的结果就是代码不够简洁！

ps:为了演示传参,这里用TextView代替Toast提示！



实现代码如下：**MyClick.java:**

```
package com.jay.example.innerlisten;

import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class MyClick implements OnClickListener {
    private TextView textshow;
    //把文本框作为参数传入
    public MyClick(TextView txt)
    {
        textshow = txt;
    }
    @Override
    public void onClick(View v) {
        //点击后设置文本框显示的文字
        textshow.setText("点击了按钮!");
    }
}
```

MainActivity.java

```
package com.jay.example.innerlisten;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.app.Activity;

public class MainActivity extends Activity {
    private Button btnshow;
    private TextView txtshow;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnshow = (Button) findViewById(R.id.btnshow);
        txtshow = (TextView) findViewById(R.id.textshow);
        //直接new一个外部类，并把TextView作为参数传入
        btnshow.setOnClickListener(new MyClick(txtshow));
    }
}
```

4）直接使用Activity作为事件监听器

> 只需要让Activity类实现XxxListener事件监听接口,在Activity中定义重写对应的事件处理器方法 eg:Actitity实现了OnClickListener接口,重写了onClick(view)方法在为某些组建添加该事件监听对象 时,直接setXxx.Listener(this)即可

实现代码如下：**MainAcivity.java:**

```
package com.jay.example.innerlisten;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
import android.app.Activity;

//让Activity方法实现OnClickListener接口
public class MainActivity extends Activity implements OnClickListen
    private Button btnshow;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnshow = (Button) findViewById(R.id.btnshow);
        //直接写个this
        btnshow.setOnClickListener(this);
    }
    //重写接口中的抽象方法
    @Override
    public void onClick(View v) {
        Toast.makeText(getApplicationContext(), "点击了按钮", Toast.
    }
}
```

5）直接绑定到标签:

> 就是直接在xml布局文件中对应得Activity中定义一个事件处理方法 eg:public void myClick(View source) source对应事件源(组件) 接着布局文件中对应要触发事件的组建,设置一个属性:onclick = "myclick"即可

实现代码如下：**MainAcivity.java:**

```
package com.jay.example.caller;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    //自定义一个方法,传入一个view组件作为参数
    public void myclick(View source)
    {
        Toast.makeText(getApplicationContext(), "按钮被点击了", Toast
    }
}
```

main.xml布局文件:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮"
        android:onClick="myclick"/>
 </LinearLayout>
```

本节小结

> 本节给大家介绍了Android中的事件处理机制，例子中的是onClickListener点击事件，当然除了这个以外还有其他的事件，比如onItemClickListener，凡是需要通过setXxxListener这些，基本上都是基于事件监听的！另外这五种方式用的比较多的是：1，2，3，5几种，看具体情况而定~

3.2 基于回调的事件处理机制

本节引言

> 在3.1中我们对Android中的一个事件处理机制——基于监听的事件处理机制进行了学习,简单的说就是 为我们的事件源(组件)添加一个监听器,然后当用户触发了事件后,交给监听器去处理,根据不同的事件 执行不同的操作;那么基于回调的事件处理机制又是什么样的原理呢？好吧，还有一个问题:你知道 什么是方法回调吗？知道吗？相信很多朋友都是了解，但又说不出来吧！好了，带着这些疑问我们 对android事件处理机制中的回调事件处理机制进行解析吧！

1.什么是方法回调？

文字表述：

> 答:是将功能定义与功能分开的一种手段,一种解耦合的设计思想;在Java中回调是通过接口来实现的, 作为一种系统架构,必须要有自己的运行环境,且需要为用户提供实现接口;实现依赖于客户,这样就可以 达到接口统一,实现不同,系统通过在不同的状态下"回调"我们的实现类,从而达到接口和实现的分离！

举个简单例子：

> 比如：你周五放学回家,你问你老妈煮好饭没,你妈说还没煮;然后你跟她说: 老妈,我看下喜羊羊,你煮好饭叫我哈! 分析:你和老妈约定了一个接口,你通过这个接口叫老妈煮饭,当饭煮好了的时候,你老妈 又通过这个接口来反馈你,"饭煮好了"！

2.Android回调的事件处理机制详解：

在Android中基于回调的事件处理机制使用场景有两个：

1）自定义view

> 当用户在GUI组件上激发某个事件时,组件有自己特定的方法会负责处理该事件 通常用法:继承基本的GUI组件,重写该组件的事件处理方法,即自定义view 注意:在xml布局中使用自定义的view时,需要使用"全限定类名"

常见**View**组件的回调方法：

android为GUI组件提供了一些事件处理的回调方法,以View为例,有以下几个方法

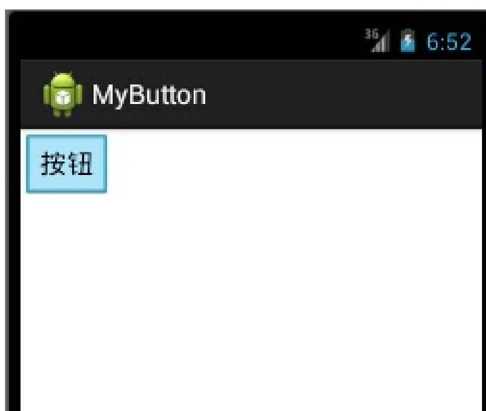
> ①在该组件上触发屏幕事件: boolean **onTouchEvent**(MotionEvent event); ②在该组件上按下某个按钮时: boolean **onKeyDown**(int keyCode,KeyEvent event); ③松开组件上的某个按钮时: boolean **onKeyUp**(int keyCode,KeyEvent event); ④长按组件某个按钮时: boolean **onKeyLongPress**(int keyCode,KeyEvent event); ⑤键盘快捷键事件发生: boolean **onKeyShortcut**(int keyCode,KeyEvent event); ⑥在组件上触发轨迹球屏事件: boolean **onTrackballEvent**(MotionEvent event); ⑦当组件的焦点发生改变,和前面的6个不同,这个方法只能够在View中重写哦！ *protected void *onFocusChanged*(boolean gainFocus, int direction, Rect previously FocusedRect)

另外，这了解释下什么是轨迹球，不过用处不大,在以前黑莓的手机上可以看到;当我们浏览网页的的时候 ,可以把轨迹球看作鼠标,不过这样的操作,我们用onTouchEvent就可以解决了,而且不够美观,所以现在 用的很好,基本不用,如果你有兴趣想看看的话,可以在原始Android模拟器按f6就可以看到了！



代码示例： 我们自定义一个MyButton类继承Button类,然后重写onKeyLongPress方法; 接着在xml文件中通过全限定类名调用自定义的view

效果图如下：



一个简单的按钮,点击按钮后触发onTouchEvent事件,当我们按模拟器上的键盘时,按下触发onKeyDown,离开键盘时触发onKeyUp事件！我们通过Logcat进行查看！

```
logcat
08-05 06:55:31.950    1064-1064/example.jay.com.mybutton I/呵呵: onTouchEvent方法被调用
08-05 06:55:32.030    1064-1064/example.jay.com.mybutton I/呵呵: onTouchEvent方法被调用
08-05 06:55:38.480    1064-1064/example.jay.com.mybutton I/呵呵: onKeyDown方法被调用
08-05 06:55:38.570    1064-1064/example.jay.com.mybutton I/呵呵: onKeyUp方法被调用
```

实现代码： MyButton.java

```
public class MyButton extends Button{ private static String
```

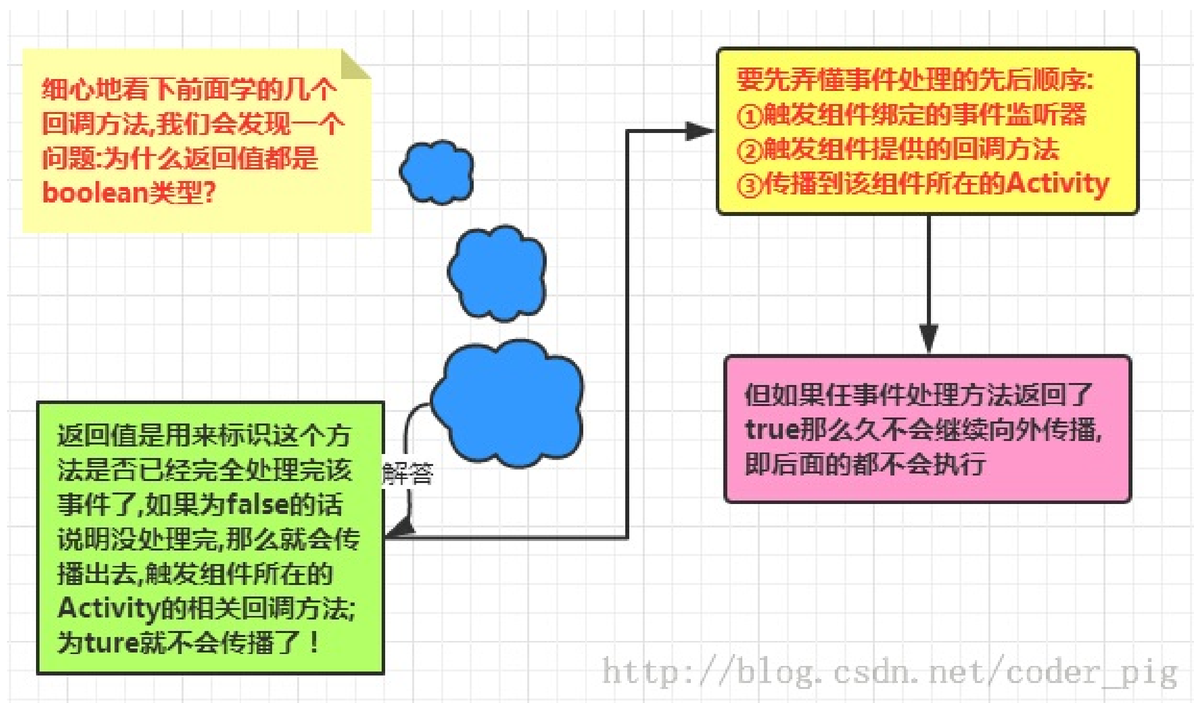
布局文件：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
```

代码解析：

> 因为我们直接重写了Button的三个回调方法,当发生点击事件后就不需要我们在Java文件中进行 事件监听器的绑定就可以完成回调,即组件会处理对应的事件,即事件由事件源(组件)自身处理！

2）基于回调的事件传播：



综上,就是如果是否向外传播取决于方法的返回值是时true还是false;

代码示例：

```
public class MyButton extends Button{ private static String
```

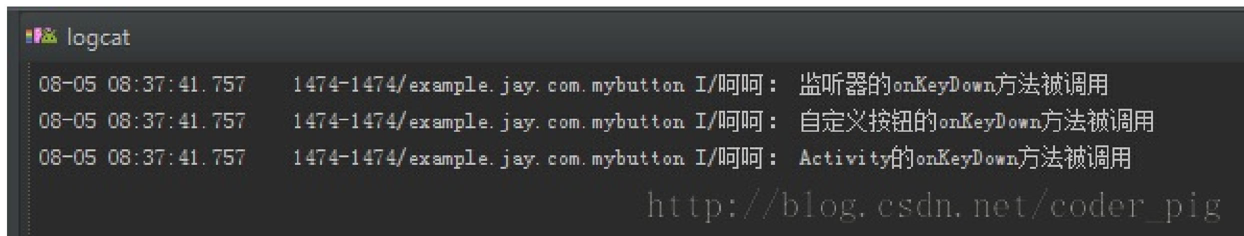
main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/ar
```

MainActivity.java：

```
public class MyActivity extends ActionBarActivity { @Override
```

运行截图：



结果分析： 从上面的运行结果,我们就可以知道,传播的顺序是: 监听器--->**view**组件的回调方法--->**Activity**的回调方法了;

本节小结

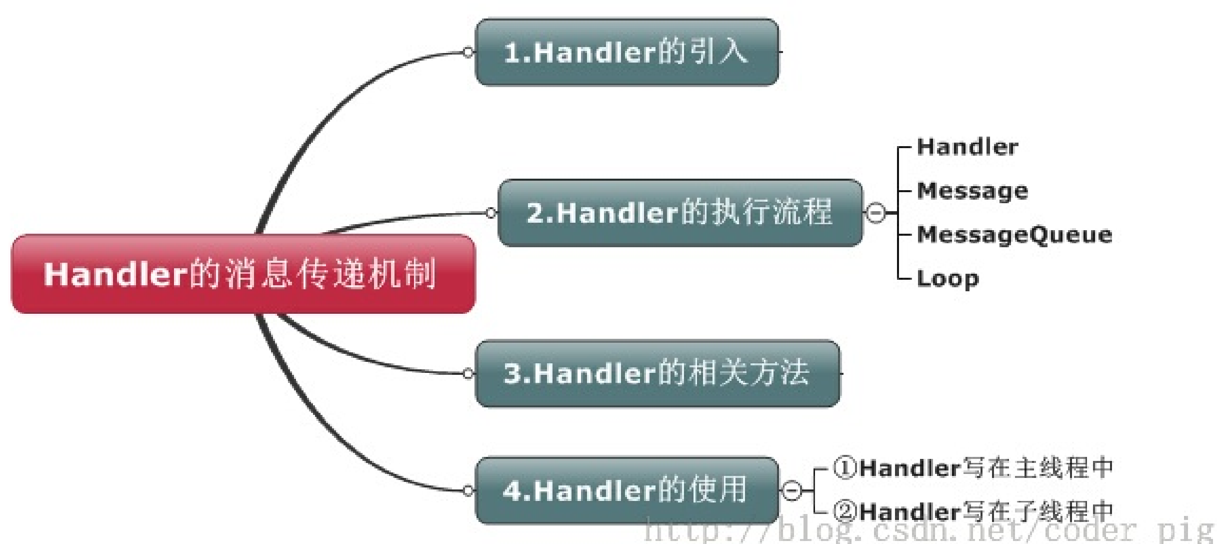
> 本节对Android事件处理机制中的基于回调的事件处理机制进行了讲解！核心就是事件传播的顺序 监听器优先，然后到View组件自身，最后再到Activity；返回值false继续传播，true终止传播~！

3.3 Handler消息传递机制浅析

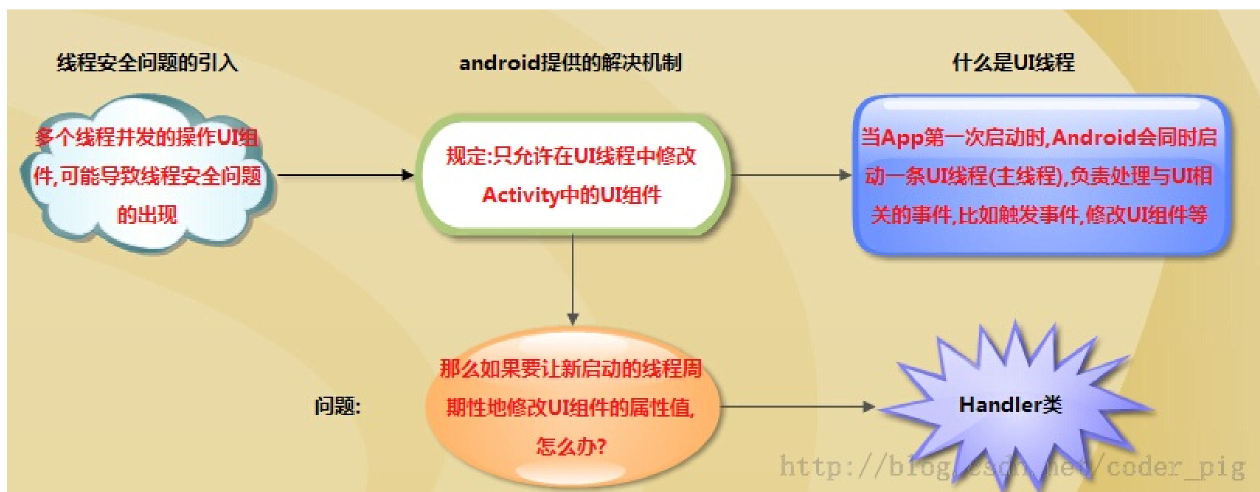
本节引言

前两节中我们对Android中的两种事件处理机制进行了学习，关于响应的事件响应就这两种；本节给大家讲解的 是Activity中UI组件中的信息传递Handler，相信很多朋友都知道，Android为了线程安全，并不允许我们在UI线程外操作UI；很多时候我们做界面刷新都需要通过Handler来通知UI组件更新！除了用Handler完成界面更新外，还可以使用runOnUiThread()来更新，甚至更高级的事务总线，当然，这里我们只讲解Handler，什么是Handler，执行流程，相关方法，子线程与主线程中中使用Handler的区别等！

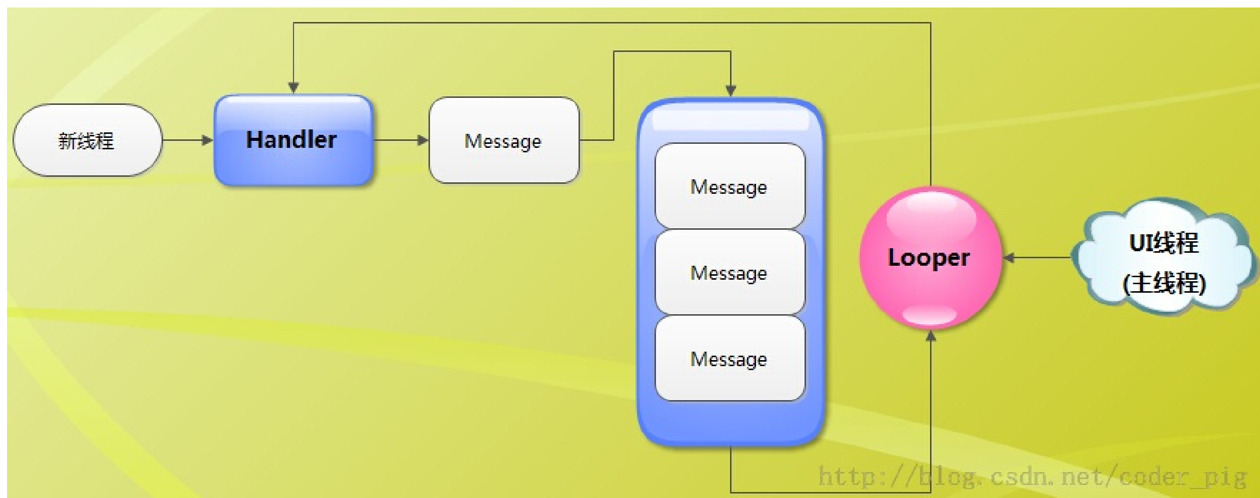
1.学习路线图：



2.Handler类的引入:



3.Handler的执行流程图：



流程图解析: 相关名词

> - **UI**线程:就是我们的主线程,系统在创建UI线程的时候会初始化一个Looper对象,同时也会创建一个与其关联的MessageQueue;
> - **Handler**:作用就是发送与处理信息,如果希望Handler正常工作,在当前线程中要有一个Looper对象
> - **Message**:Handler接收与处理的消息对象
> - **MessageQueue**:消息队列,先进先出管理Message,在初始化Looper对象时会创建一个与之关联的MessageQueue;
> - **Looper**:每个线程只能够有一个Looper,管理MessageQueue,不断地从中取出Message分发给对应的Handler处理！

简单点说：

> 当我们的子线程想修改Activity中的UI组件时,我们可以新建一个Handler对象,通过这个对象向主线程发送信息;而我们发送的信息会先到主线程的MessageQueue进行等待,由Looper按先入先出顺序取出,再根据message对象的what属性分发给对应的Handler进行处理！

4.Handler的相关方法:

> - void **handleMessage**(Message msg):处理消息的方法,通常是用于被重写!
> - **sendEmptyMessage**(int what):发送空消息
> - **sendEmptyMessageDelayed**(int what,long delayMillis):指定延时多少毫秒后发送空信息
> - **sendMessage**(Message msg):立即发送信息
> - **sendMessageDelayed**(Message msg):指定延时多少毫秒后发送信息
> - final boolean **hasMessage**(int what):检查消息队列中是否包含what属性为指定值的消息 如果是参数为(int what,Object object):除了判断what属性,还需要判断Object属性是否为指定对象的消息

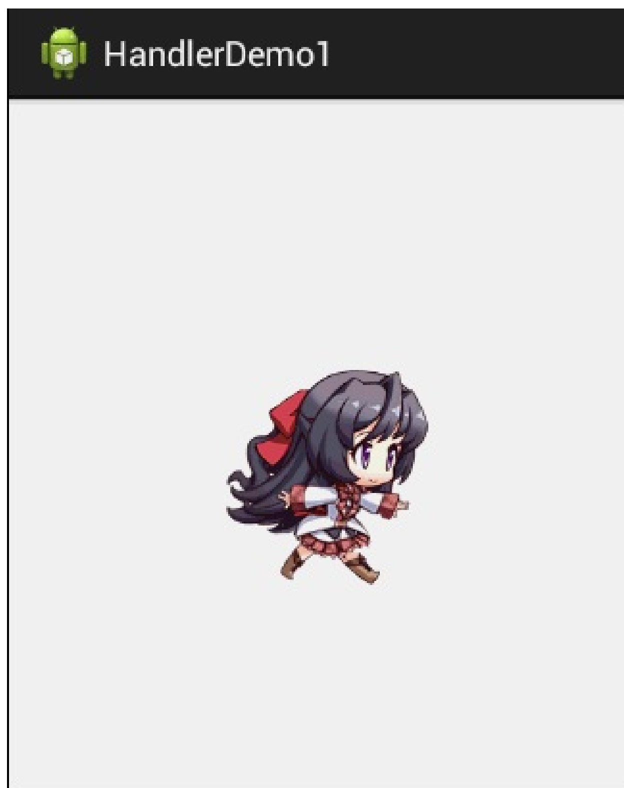
5.Handler的使用示例：

1）Handler写在主线程中

> 在主线程中,因为系统已经初始化了一个Looper对象,所以我们直接创建Handler对象,就可以进行信息的发送与处理了！

代码示例： 简单的一个定时切换图片的程序,通过Timer定时器,定时修改ImageView显示的内容,从而形成帧动画

运行效果图：



实现代码：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context="com.jay.example.handlerdemo1.MainActivity" >

    <ImageView
        android:id="@+id/imgchange"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />

</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends Activity {

    //定义切换的图片的数组id
    int imgids[] = new int[]{
        R.drawable.s_1, R.drawable.s_2,R.drawable.s_3,
        R.drawable.s_4,R.drawable.s_5,R.drawable.s_6,
        R.drawable.s_7,R.drawable.s_8
    };
    int imgstart = 0;

    final Handler myHandler = new Handler()
    {
      @Override
      //重写handleMessage方法,根据msg中what的值判断是否执行后续操作
      public void handleMessage(Message msg) {
        if(msg.what == 0x123)
           {
            imgchange.setImageResource(imgids[imgstart++ % 8]);
           }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final ImageView imgchange = (ImageView) findViewById(R.id.i

        //使用定时器,每隔200毫秒让handler发送一个空信息
        new Timer().schedule(new TimerTask() {
            @Override
            public void run() {
                myHandler.sendEmptyMessage(0x123);

            }
        }, 0,200);
    }

}
```

2）Handler写在子线程中

如果是Handler写在了子线程中的话,我们就需要自己创建一个Looper对象了!创建的流程如下:

> **1)**直接调用Looper.prepare()方法即可为当前线程创建Looper对象,而它的构造器会创建配套的MessageQueue; **2)**创建Handler对象,重写handleMessage( )方法就可以处理来自于其他线程的信息了! **3)**调用Looper.loop()方法启动Looper

使用示例： 输入一个数，计算后通过Toast输出在这个范围内的所有质数

实现代码： **main.xml**：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/etNum"
        android:inputType="number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="请输入上限"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="cal"
        android:text="计算"/>
</LinearLayout>
```

MainActivity.java:

```
public class CalPrime extends Activity
{
    static final String UPPER_NUM = "upper";
    EditText etNum;
    CalThread calThread;
    // 定义一个线程类
    class CalThread extends Thread
    {
        public Handler mHandler;

        public void run()
        {
            Looper.prepare();
            mHandler = new Handler()
            {
                // 定义处理消息的方法
                @Override
                public void handleMessage(Message msg)
                {
                    if(msg.what == 0x123)
                    {
                        int upper = msg.getData().getInt(UPPER_NUM)
                        List<Integer> nums = new ArrayList<Integer>
                        // 计算从2开始、到upper的所有质数
                        outer:
                        for (int i = 2 ; i <= upper ; i++)
                        {
```

```
                        // 用i处于从2开始、到i的平方根的所有数
                        for (int j = 2 ; j <= Math.sqrt(i) ; j+
                        {
                            // 如果可以整除，表明这个数不是质数
                            if(i != 2 && i % j == 0)
                            {
                                continue outer;
                            }
                        }
                        nums.add(i);
                    }
                    // 使用Toast显示统计出来的所有质数
                    Toast.makeText(CalPrime.this , nums.toStrin
                        , Toast.LENGTH_LONG).show();
                }
            }
        };
        Looper.loop();
    }
}
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    etNum = (EditText)findViewById(R.id.etNum);
    calThread = new CalThread();
    // 启动新线程
    calThread.start();
}
// 为按钮的点击事件提供事件处理函数
public void cal(View source)
{
    // 创建消息
    Message msg = new Message();
    msg.what = 0x123;
    Bundle bundle = new Bundle();
    bundle.putInt(UPPER_NUM ,
        Integer.parseInt(etNum.getText().toString()));
    msg.setData(bundle);
    // 向新线程中的Handler发送消息
    calThread.mHandler.sendMessage(msg);
}
}
```

PS:本例子来自于《Android疯狂讲义》~

本节小结

本节对Android中的Handler事件传递进行了简单的分析，要分清楚Handler，Message，MessageQueue， Loop的概念，以及Handler写在主线程中以及子线程中的区别！

3.4 TouchListener PK OnTouchEvent + 多点触碰

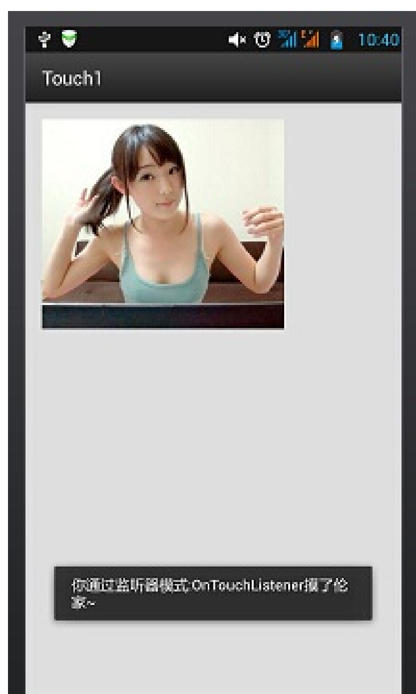
本节引言：

> 如题，本节给大家带来的是**TouchListener**与**OnTouchEvent**的比较，以及多点触碰的知识点！ TouchListener是基于监听的，而OnTouchEvent则是基于回调的！下面通过两个简单的例子来加深 大家的理解！

1.基于监听的TouchListener

代码示例：

实现效果图：



实现代码： **main.xml**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MyActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imgtouch"
        android:background="@drawable/touch"/>
</RelativeLayout>
```

MainAcitivity.java

```
public class MyActivity extends ActionBarActivity {

    private ImageView imgtouch;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);

        imgtouch = (ImageView)findViewById(R.id.imgtouch);
        imgtouch.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                Toast.makeText(getApplicationContext(),"你通过监听器
                return true;
            }
        });
    }
}
```

代码解析：

> 就是简单的设置一个ImageView,然后setOnTouchListener,重写onTouch方法即可!很简单,其实这个在帧布局那一节已经有个例子了，还记得那个随手指移动的萌妹子吗？

OnTouchListener相关方法与属性:

> **onTouch**(View v, MotionEvent event):这里面的参数依次是触发触摸事件的组件,触碰事件event 封装了触发事件的详细信息，同样包括事件的类型、触发时间等信息。比如event.getX(),event.getY() 我们也可以对触摸的动作类型进行判断,使用event.getAction( )再进行判断;如: event.getAction == MotionEvent.**ACTION_DOWN**：按下事件 event.getAction == MotionEvent.**ACTION_MOVE**:移动事件 event.getAction == MotionEvent.**ACTION_UP**:弹起事件

2.基于回调的onTouchEvent( )方法

> 同样是触碰事件,但是onTouchEvent更多的是用于自定义的view,所有的view类中都重写了该方法,而这种触摸事件是基于回调的,也就是说:如果我们返回的值是false的话,那么事件会继续向外传播,由外面的容器或者Activity进行处理!当然还涉及到了手势(Gesture),这个我们会在后面进行详细的讲解!onTouchEvent其实和onTouchListener是类似的,只是处理机制不用,前者是回调,后者是监听模式！

代码示例： 定义一个简单的view，绘制一个蓝色的小圆，可以跟随手指进行移动

实现代码： MyView.java

```
public class MyView extends View{
    public float X = 50;
    public float Y = 50;

    //创建画笔
    Paint paint = new Paint();

    public MyView(Context context,AttributeSet set)
    {
        super(context,set);
    }

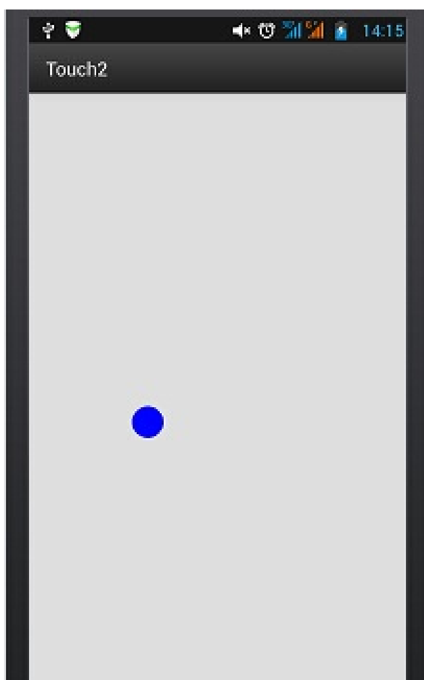
    @Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        paint.setColor(Color.BLUE);
        canvas.drawCircle(X,Y,30,paint);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        this.X = event.getX();
        this.Y = event.getY();
        //通知组件进行重绘
        this.invalidate();
        return true;
    }
}
```

main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MyActivity">
    <example.jay.com.touch2.MyView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

实现效果图：



用手指触摸进行移动~

3.多点触碰

原理类的东西：

所谓的多点触碰就是多个手指在屏幕上进行操作，用的最多的估计是放大缩功能吧，比如很多的图片浏览器都支持缩放！理论上Android系统本身可以处理多达256个手指的触摸，当然这取决于手机硬件的支持；不过支持多点触摸的手机一般支持2-4个点，当然有些更多！我们发现前面两点都有用到MotionEvent，MotionEvent代表的是一个触摸事件；前我们可以根据**event.getAction() & MotionEvent.ACTION_MASK**来判断是哪种操作，除了上面介绍的三种单点操作外，还有两个多点专用的操作：

- MotionEvent.**ACTION_POINTER_DOWN**:当屏幕上已经有一个点被按住，此时再按下其他点时触发。

- MotionEvent.**ACTION_POINTER_UP**:当屏幕上有多个点被按住，松开其中一个点时触发（即非最后一个点被放开时）。

简单的流程大概是这样：

- 当我们一个手指触摸屏幕 ——> 触发ACTION_DOWN事件
- 接着有另一个手指也触摸屏幕 ——> 触发ACTION_POINTER_DOWN事件,如果还有其他手指触摸，继续触发
- 有一个手指离开屏幕 ——> 触发ACTION_POINTER_UP事件，继续有手指离开，继续触发
- 当最后一个手指离开屏幕 ——> 触发ACTION_UP事件
- 而且在整个过程中，ACTION_MOVE事件会一直不停地被触发

我们可以通过event.**getX**(int)或者event.**getY**(int)来获得不同触摸点的位置： 比如event.getX(0)可以获得第一个接触点的X坐标，event.getX(1)获得第二个接触点的X坐标这样... 另外，我们还可以在调用MotionEvent对象的**getPointerCount()**方法判断当前有多少个手指在触摸~

代码示例：

好吧，我们来写个最常见的单指拖动图片，双指缩放图片的示例吧：

实现效果图：



实现代码：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/img_test"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="matrix"
        android:src="@drawable/pic1" />

</RelativeLayout>
```

MainActivity.java

```
package com.jay.example.edittextdemo;

import android.app.Activity;
import android.graphics.Matrix;
import android.graphics.PointF;
import android.os.Bundle;
import android.util.FloatMath;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;

public class MainActivity extends Activity implements OnTouchLister

    private ImageView img_test;

    // 縮放控制
    private Matrix matrix = new Matrix();
    private Matrix savedMatrix = new Matrix();

    // 不同状态的表示：
    private static final int NONE = 0;
    private static final int DRAG = 1;
    private static final int ZOOM = 2;
    private int mode = NONE;

    // 定义第一个按下的点，两只接触点的重点，以及出事的两指按下的距离：
    private PointF startPoint = new PointF();
    private PointF midPoint = new PointF();
    private float oriDis = 1f;

    /*
     * (non-Javadoc)
     *
```

```
     * @see android.app.Activity#onCreate(android.os.Bundle)
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img_test = (ImageView) this.findViewById(R.id.img_test);
        img_test.setOnTouchListener(this);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        ImageView view = (ImageView) v;
        switch (event.getAction() & MotionEvent.ACTION_MASK) {
        // 单指
        case MotionEvent.ACTION_DOWN:
            matrix.set(view.getImageMatrix());
            savedMatrix.set(matrix);
            startPoint.set(event.getX(), event.getY());
            mode = DRAG;
            break;
        // 双指
        case MotionEvent.ACTION_POINTER_DOWN:
            oriDis = distance(event);
            if (oriDis > 10f) {
                savedMatrix.set(matrix);
                midPoint = middle(event);
                mode = ZOOM;
            }
            break;
        // 手指放开
        case MotionEvent.ACTION_UP:
        case MotionEvent.ACTION_POINTER_UP:
            mode = NONE;
            break;
        // 单指滑动事件
        case MotionEvent.ACTION_MOVE:
            if (mode == DRAG) {
                // 是一个手指拖动
                matrix.set(savedMatrix);
                matrix.postTranslate(event.getX() - startPoint.x, e
            } else if (mode == ZOOM) {
                // 两个手指滑动
                float newDist = distance(event);
                if (newDist > 10f) {
                    matrix.set(savedMatrix);
                    float scale = newDist / oriDis;
                    matrix.postScale(scale, scale, midPoint.x, midP
                }
            }
            break;
        }
        // 设置ImageView的Matrix
```

```
            view.setImageMatrix(matrix);
            return true;
        }

        // 计算两个触摸点之间的距离
        private float distance(MotionEvent event) {
            float x = event.getX(0) - event.getX(1);
            float y = event.getY(0) - event.getY(1);
            return FloatMath.sqrt(x * x + y * y);
        }

        // 计算两个触摸点的中点
        private PointF middle(MotionEvent event) {
            float x = event.getX(0) + event.getX(1);
            float y = event.getY(0) + event.getY(1);
            return new PointF(x / 2, y / 2);
        }

    }
```

本节小结：

好的，关于**TouchListener****和OnTouchEvent**以及多点触碰就到这里~

3.5 监听EditText的内容变化

本节引言：

在前面我们已经学过EditText控件了，本节来说下如何监听输入框的内容变化！ 这个再实际开发中非常实用，另外，附带着说下如何实现EditText的密码可见与不可见！好了，开始本节内容！

1.监听EditText的内容变化

> 由题可知，是基于监听的事件处理机制，好像前面的点击事件是OnClickListener，文本内容 变化的监听器则是：TextWatcher，我们可以调用EditText.addTextChangedListener(mTextWatcher); 为EditText设置内容变化监听！

简单说下TextWatcher，实现该类需实现三个方法：

```
public void beforeTextChanged(CharSequence s, int start,int count,
public void onTextChanged(CharSequence s, int start, int before, in
public void afterTextChanged(Editable s);
```

依次会在下述情况中触发：

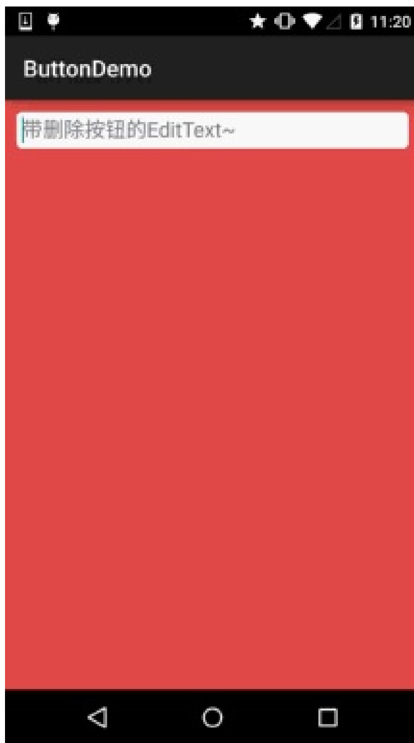
- 1.内容变化前
- 2.内容变化中
- 3.内容变化后

我们可以根据实际的需求重写相关方法，一般重写得较多的是第三个方法！

监听EditText内容变化的场合有很多： 限制字数输入，限制输入内容等等~

这里给大家实现一个简单的自定义EditText，输入内容后，有面会显示一个叉叉的圆圈，用户点击后 可以清空文本框~，当然你也可以不自定义，直接为EditText添加TextWatcher然后设置下删除按钮~

实现效果图：



自定义EditText：**DelEditText.java**

```
package demo.com.jay.buttondemo;

import android.content.Context;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.widget.EditText;

/**
 * Created by coder-pig on 2015/7/16 0016.
 */
public class DelEditText extends EditText {

    private Drawable imgClear;
    private Context mContext;

    public DelEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.mContext = context;
        init();
    }

    private void init() {
        imgClear = mContext.getResources().getDrawable(R.drawable.d
        addTextChangedListener(new TextWatcher() {
            @Override
```

```
            public void beforeTextChanged(CharSequence s, int start

            }

            @Override
            public void onTextChanged(CharSequence s, int start, in

            }

            @Override
            public void afterTextChanged(Editable editable) {
                setDrawable();
            }
        });
    }

    //绘制删除图片
    private void setDrawable(){
        if (length() < 1)
            setCompoundDrawablesWithIntrinsicBounds(null, null, nul
        else
            setCompoundDrawablesWithIntrinsicBounds(null, null, img
    }

    //当触摸范围在右侧时，触发删除方法，隐藏叉叉
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if(imgClear != null && event.getAction() == MotionEvent.AC
        {
            int eventX = (int) event.getRawX();
            int eventY = (int) event.getRawY();
            Rect rect = new Rect();
            getGlobalVisibleRect(rect);
            rect.left = rect.right - 100;
            if (rect.contains(eventX, eventY))
                setText("");
        }
        return super.onTouchEvent(event);
    }

    @Override
    protected void finalize() throws Throwable {
        super.finalize();
    }

}
```

EditText的背景drawable：**bg_frame_search.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >
    <solid android:color="@color/background_white" />
    <corners android:radius="5dp" />
    <stroke android:width="1px" android:color="@color/frame_search'
</shape>
```

颜色资源:color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="reveal_color">#FFFFFF</color>
    <color name="bottom_color">#3086E4</color>
    <color name="bottom_bg">#40BAF8</color>
    <color name="frame_search">#ADAEAD</color>
    <color name="background_white">#FFFFFF</color>
    <color name="back_red">#e75049</color>
</resources>
```

布局文件：**activity_main.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/back_red"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <demo.com.jay.buttondemo.DelEditText
        android:id="@+id/edit_search"
        android:layout_width="match_parent"
        android:layout_height="32dp"
        android:layout_margin="10dp"
        android:background="@drawable/bg_frame_search"
        android:hint="带删除按钮的EditText~"
        android:maxLength="20"
        android:padding="5dp"
        android:singleLine="true" />

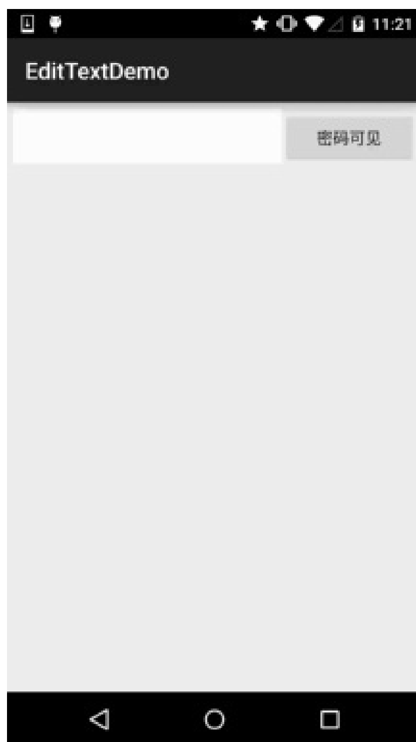
</LinearLayout>
```

PS:代码是非常简单的，就不解释了~

2.实现EditText的密码可见与不可见

这个也是一个很实用的需求，就是用户点击按钮后可让EditText中的密码可见或者不可见~

实现效果图：



实现代码： **activity_main.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:layout_margin="5dp"
    android:orientation="horizontal">

    <EditText
        android:id="@+id/edit_pawd"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="48dp"
        android:inputType="textPassword"
        android:background="@drawable/editborder"/>

    <Button
        android:id="@+id/btnChange"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="48dp"
        android:text="密码可见"/>

</LinearLayout>
```

MainActivity.java

```
package com.jay.demo.edittextdemo;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    private EditText edit_pawd;
    private Button btnChange;
    private boolean flag = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        edit_pawd = (EditText) findViewById(R.id.edit_pawd);
        btnChange = (Button) findViewById(R.id.btnChange);
        edit_pawd.setHorizontallyScrolling(true);    //设置EditText
        btnChange.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(flag == true){
                    edit_pawd.setTransformationMethod(HideReturnsTr
                    flag = false;
                    btnChange.setText("密码不可见");
                }else{
                    edit_pawd.setTransformationMethod(PasswordTrans
                    flag = true;
                    btnChange.setText("密码可见");
                }
            }
        });
    }
}
```

editborder.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" >

    <!-- 设置透明背景色 -->
    <solid android:color="#FFFFFF" />

    <!-- 设置一个白色边框 -->
    <stroke
        android:width="1px"
        android:color="#FFFFFF" />
    <!-- 设置一下边距,让空间大一点 -->
    <padding
        android:bottom="5dp"
        android:left="5dp"
        android:right="5dp"
        android:top="5dp" />

</shape>
```

本节小结：

本节就到这里，谢谢~

3.6 响应系统设置的事件(Configuration类)

本节引言：

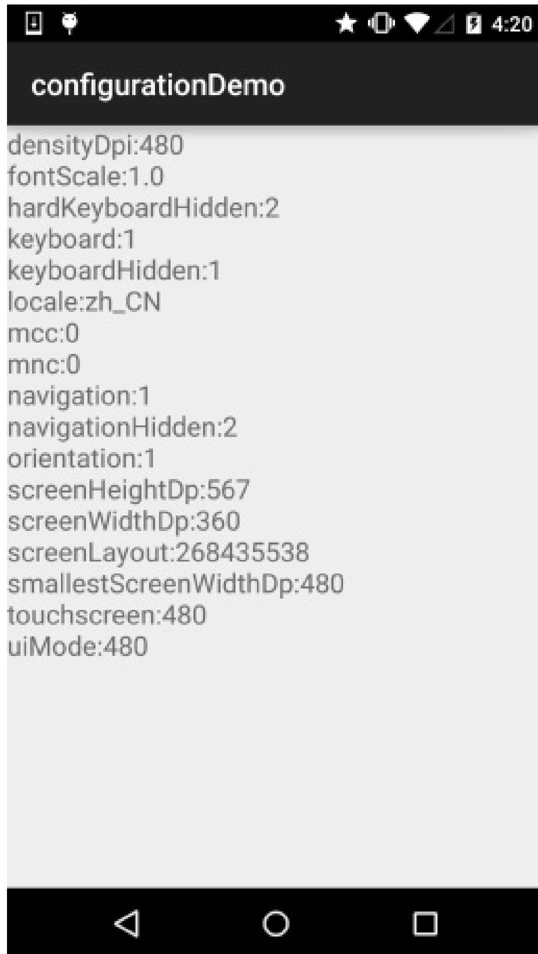
> 本节给大家介绍的Configuration类是用来描述手机设备的配置信息的，比如屏幕方向， 触摸屏的触摸方式等，相信定制过ROM的朋友都应该知道我们可以在: frameworks/base/core/java/android/content/res/Configuration.java 找到这个类，然后改下相关设置，比如调整默认字体的大小！有兴趣可自行了解！ 本节讲解的Configuration类在我们Android开发中的使用~ API文档：Configuration

1.Configuration给我们提供的方法列表

> - **densityDpi**：屏幕密度
> - **fontScale**：当前用户设置的字体的缩放因子
> - **hardKeyboardHidden**：判断硬键盘是否可见，有两个可选值：HARDKEYBOARDHIDDEN_NO,HARDKEYBOARDHIDDEN_YES，分别是十六进制的0和1
> - **keyboard**：获取当前关联额键盘类型：该属性的返回值：KEYBOARD_12KEY（只有12个键的小键盘）、KEYBOARD_NOKEYS、KEYBOARD_QWERTY（普通键盘）
> - **keyboardHidden**：该属性返回一个boolean值用于标识当前键盘是否可用。该属性不仅会判断系统的硬件键盘，也会判断系统的软键盘（位于屏幕）。
> - **locale**：获取用户当前的语言环境
> - **mcc**：获取移动信号的国家码
> - **mnc**：获取移动信号的网络码 ps:国家代码和网络代码共同确定当前手机网络运营商
> - **navigation**：判断系统上方向导航设备的类型。该属性的返回值：NAVIGATION_NONAV（无导航）、 NAVIGATION_DPAD(DPAD导航） NAVIGATION_TRACKBALL（轨迹球导航）、NAVIGATION_WHEEL（滚轮导航）
> - **orientation**：获取系统屏幕的方向。该属性的返回值：ORIENTATION_LANDSCAPE（横向屏幕）、ORIENTATION_PORTRAIT（竖向屏幕）
> - **screenHeightDp**，**screenWidthDp**：屏幕可用高和宽，用dp表示
> - **touchscreen**：获取系统触摸屏的触摸方式。该属性的返回值：TOUCHSCREEN_NOTOUCH（无触摸屏）、TOUCHSCREEN_STYLUS（触摸笔式触摸屏）、TOUCHSCREEN_FINGER（接收手指的触摸屏）

2.写个简单例子测试下：

运行截图：



代码实现：

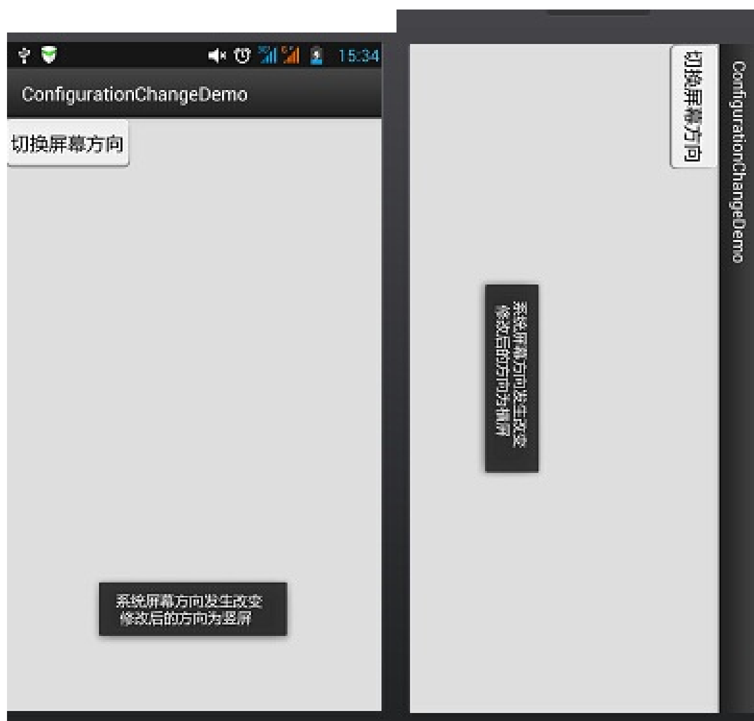
```
public class MainActivity extends AppCompatActivity { @Overr
```

3.重写onConfigurationChanged响应系统设置更改

> 该方法用于监听系统设置的更改,是基于回调的时间处理方法,当系统的设置发生改变时就会自动触发; 但是要注意一点,使用下面的方法监控的话,targetSdkVersion属性最高只能设置为12,高于12的话,该方法不会被激发！这里写个横竖屏切换的例子给大家参考参考，其他的可自行谷歌~

代码示例：简单的一个按钮,点击后切换横竖屏,然后Toast提示

运行效果图：



实现代码：

```
public class MainActivity extends Activity { @Override prote
```

另外，还需要在AndroidManifest.xml添加下述内容：

> 权限: **< uses-permission android:name="android.permission.CHANGE_CONFIGURATION" />** 在< activity标签中添加:**android:configChanges="orientation"** 将targetSdkVersion改为12以上的,12也可以

本节小结：

本节给大家讲解了：Configuration类以及onConfigurationChanged响应系统设置更改，有个大概了解即可 后续用到我们再继续深入~

3.7 AnsyncTask异步任务

本节引言：

本节给大家带来的是Android给我们提供的一个轻量级的用于处理异步任务的类:AsyncTask，我们一般是 继承AsyncTask，然后在类中实现异步操作，然后将异步执行的进度，反馈给UI主线程~ 好吧，可能有些概念大家不懂，觉得还是有必要讲解下多线程的概念，那就先解释下一些概念性的东西吧！

1.相关概念

1）什么是多线程：

> 答：先要了解这几个名称：应用程序，进程，线程，多线程！！
>
> - 应用程序(**Application**)：为了完成特定任务，用某种语言编写的一组指令集合(一组静态代码)
> - 进程(**Process**) :运行中的程序，系统调度与资源分配的一个独立单位，操作系统会为每个进程分配 一段内存空间，程序的依次动态执行，经理代码加载 -> 执行 -> 执行完毕的完整过程！
> - 线程(**Thread**)：比进程更小的执行单元，每个进程可能有多条线程，线程需要放在一个进程中才能执行！ 线程是由程序负责管理的！！！而进程则是由系统进行调度的！！！
> - 多线程概念(**Multithreading**)：并行地执行多条指令，将CPU的时间片按照调度算法，分配给各个线程，实际上是分时执行的，只是这个切换的时间很短，用户感觉是同时而已！

举个简单的例子： 你挂着QQ，突然想去听歌，你需要把QQ关掉，然后再去启动XX播放器吗？答案是否定的，我们直接打开播放器 放歌就好，QQ还在运行着，是吧！这就是简单的多线程~在实际开发中，也有这样的例子，比如应用正在运行，发现新版本了，想后台更新，这个时候一般我们会开辟出一条后台线程，用于下载新版本的apk，但是这个时候 我们还可以使用应用中的其他功能！这就是多线程的使用例子~

2）同步与异步的概念：

> 答: 同步：当我们执行某个功能时，在没有得到结果之前，这个调用就不能返回！简单点就是说必须 等前一件事做完才能做下一件事；举个简单的例子：比如你啪啪啪，为了避免弄出人命，肯定要先戴好套套， 然后再啪啪啪是吧~套套戴好 -> 然后啪啪啪，比如你没套套，那啪啪啪的操作就要等待了，直到你把套套买回来带上，这个时候就可以开始啪啪啪了~一个形象地例子，♪(^∇^) *异步：和同步则是相对的，当我们执行某个功能后，我们并不需要立即得到结果，我们额可以正常地 做其他操作，这个功能可以在完成后通知或者回调来告诉我们；还是上面那个后台下载的例子，后台下载， 我们执行下载功能后，我们就无需去关心它的下载过程，当下载完毕后通知我们就可以了~

3) Android为很么要引入异步任务

> 答：因为Android程序刚启动时，会同时启动一个对应的主线程(Main Thread)，这个主线程主要负责处理 与UI相关的事件！有时我们也把他称作UI线程！而在Android App时我们必须遵守这个单线程模型的规则： **Android UI操作并不是线程安全的并且这些操作都需要在UI线程中执行！** 假如我们在非UI线程中，比如在主线程中new Thread()另外开辟一个线程，然后直接在里面修改UI控件的值；此时会抛出下述异常：**android.view.ViewRoot$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views** 另外，还有一点，如果我们把耗时的操作都放在UI线程中的话，如果UI线程超过5s没有响应用于请求，那么 这个时候会引发ANR(Application Not Responding)异常，就是应用无响应~ 最后还有一点就是：Android 4.0后禁止在UI线程中执行网络操作~不然会报: **android.os.NetworkOnMainThreadException**

以上的种种原因都说明了Android引入异步任务的意义，当然实现异步也不可以不用到我们本节讲解 的AsyncTask，我们可以自己开辟一个线程，完成相关操作后，通过下述两种方法进行UI更新：

> 1. 前面我们学的Handler，我们在Handler里写好UI更新，然后通过sendMessage()等的方法通知UI 更新，另外别忘了Handler写在主线程和子线程中的区别哦~
> 2. 利用Activity.runOnUiThread(Runnable)把更新ui的代码创建在Runnable中,更新UI时，把Runnable 对象传进来即可~

2.AsyncTask全解析：

1）为什么要用AsyncTask？

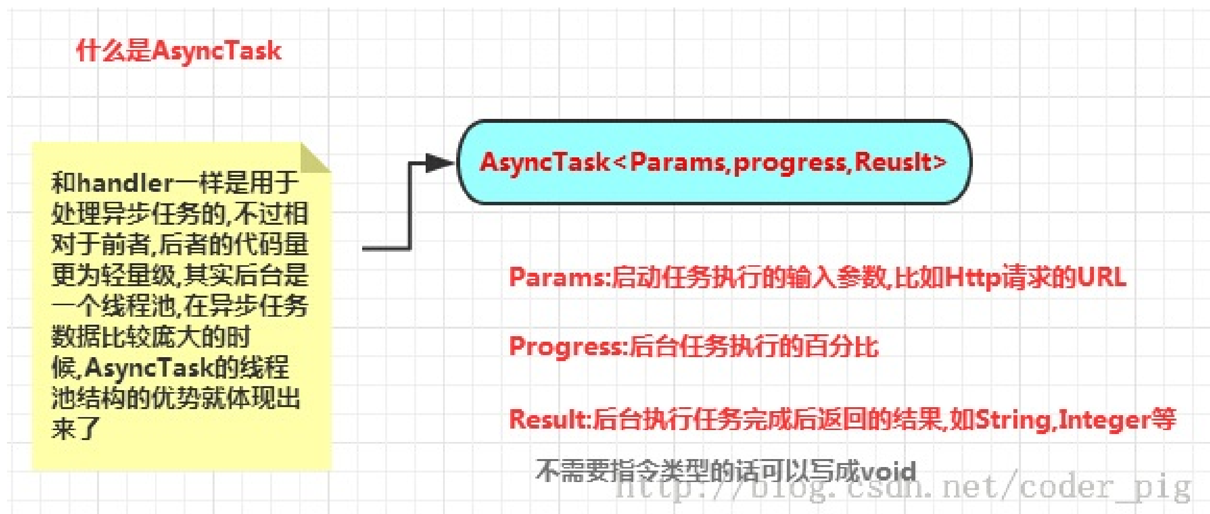
答:我们可以用上述两种方法来完成我们的异步操作，加入要我们写的异步操作比较多，或者较为繁琐， 难道我们new Thread()然后用上述方法通知UI更新么？程序员都是比较喜欢偷懒的，既然官方给我 们提供了AsyncTask这个封装好的轻量级异步类，为什么不用呢？我们通过几十行的代码就可以完成 我们的异步操作，而且进度可控；相比起Handler，AsyncTask显得更加简单，快捷~当然，这只适合 简单的异步操作，另外，实际异步用的最多的地方就是网络操作，图片加载，数据传输等，AsyncTask 暂时可以满足初学者的需求，谢谢小应用，但是到了公司真正做项目以

后，我们更多的使用第三发的 框架，比如Volley,OkHttp,android-async-http,XUtils等很多，后面进阶教程我们会选1-2个框架进行 学习，当然你可以自己找资料学习学习，但是掌握AsyncTask还是很有必要的！

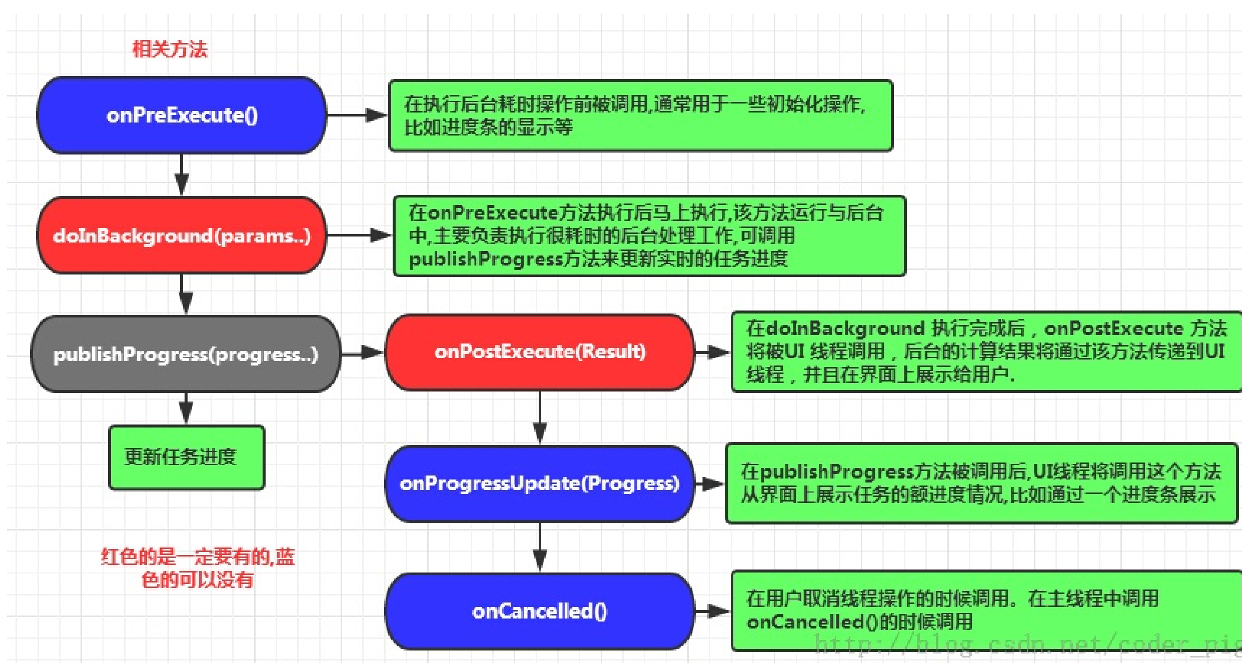
2）AsyncTask的基本结构：

AsyncTask是一个抽象类，一般我们都会定义一个类继承AsyncTask然后重写相关方法~ 官方API:AsyncTask

- 构建AsyncTask子类的参数：



- 相关方法与执行流程：



- 注意事项：

注意事项:

①Task的实例必须在UI thread中创建；
②execute方法必须在UI thread中调用；
③不要手动的调用onPreExecute(), onPostExecute(Result)，doInBackground(Params...), onProgressUpdate(Progress...)这几个方法；
④该task只能被执行一次，否则多次调用时将会出现异常；

3.AsyncTask使用示例：

> 因为我们还没学到Android网络那块，这里照顾下各位初学者，这里用延时 线程来模拟文件下载的过程~后面讲到网络那里再给大家写几个例子~

实现效果图：



布局文件:**activity.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MyActivity">
    <TextView
        android:id="@+id/txttitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <!--设置一个进度条,并且设置为水平方向-->
    <ProgressBar
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/pgbar"
        style="?android:attr/progressBarStyleHorizontal"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnupdate"
        android:text="更新progressBar"/>
</LinearLayout>
```

定义一个延时操作，用于模拟下载：

```
public class DelayOperator {
    //延时操作,用来模拟下载
    public void delay()
    {
        try {
            Thread.sleep(1000);
        }catch (InterruptedException e){
            e.printStackTrace();;
        }
    }
}
```

自定义**AsyncTask:**

```
public class MyAsyncTask extends AsyncTask<Integer,Integer,String>
{
    private TextView txt;
    private ProgressBar pgbar;

    public MyAsyncTask(TextView txt,ProgressBar pgbar)
    {
        super();
        this.txt = txt;
        this.pgbar = pgbar;
    }

    //该方法不运行在UI线程中,主要用于异步操作,通过调用publishProgress()方法
    //触发onProgressUpdate对UI进行操作
    @Override
    protected String doInBackground(Integer... params) {
        DelayOperator dop = new DelayOperator();
        int i = 0;
        for (i = 10;i <= 100;i+=10)
        {
            dop.delay();
            publishProgress(i);
        }
        return  i + params[0].intValue() + "";
    }

    //该方法运行在UI线程中,可对UI控件进行设置
    @Override
    protected void onPreExecute() {
        txt.setText("开始执行异步线程~");
    }

    //在doBackground方法中,每次调用publishProgress方法都会触发该方法
    //运行在UI线程中,可对UI控件进行操作

    @Override
    protected void onProgressUpdate(Integer... values) {
        int value = values[0];
        pgbar.setProgress(value);
    }
}
```

MainActivity.java：

```
public class MyActivity extends ActionBarActivity {

    private TextView txttitle;
    private ProgressBar pgbar;
    private Button btnupdate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txttitle = (TextView)findViewById(R.id.txttitle);
        pgbar = (ProgressBar)findViewById(R.id.pgbar);
        btnupdate = (Button)findViewById(R.id.btnupdate);
        btnupdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MyAsyncTask myTask = new MyAsyncTask(txttitle,pgbar
                myTask.execute(1000);
            }
        });
    }
}
```

本节小结：

> 好的，本节一开始给大家普及了下应用程序，进程，线程，多线程，异步，同步的概念；接着又讲解 了下Android中为何要引入异步操作，然后介绍了下AsyncTask的用法，当然上面也说了，异步操作在网络 操作用的较多，后面在讲解网络操作时会用到这个AsyncTask，敬请期待~本节就到这里，谢谢~

3.8 Gestures(手势)

本节引言：

周六不休息，刚剪完了个大平头回来，继续码字~

好的，本节给大家带来点的是第三章的最后一节——Gestures(手势)， 用过魅族手机的朋友相信对手势肯定是不陌生的，在home键两侧像屏幕内滑动， 可以打开后台任务列表等等~在应用中通过手势来操作会大大提升用户体验， 比如Scroll手势在浏览器中个滚屏，Fling在浏览器中的换页等！

当然，有利也有弊，比如不当的手势操作引起APP Carsh，经常这样可是会引起用户不满的！ 所以是否要为你的应用增加手势，可要考虑清楚哦！另外手势要和前面学的单指/多指触碰相区分哦！

手势是:连续触碰的行为，比如左右上下滑动屏幕，又或者画一些不规则的几何图形！ Android对上述两种手势行为都提供了支持：

- Android提供手势检测，并为手势识别提供了相应的监听器！
- Android运行开发者自行添加手势，并且提供了相应的API识别用户手势！

如果你的手机是Android 4.x的原生Android系统的话，你可能可以在你的手机或者平板上看到谷歌 提供的一个Gesture Builder的APP，该应用允许用户以类似于涂鸦的方式绘制一个手写符号，使之 对应一个字符串名称！当然，没有这样的手机也没关系，我们有模拟器嘛，自己开个4.0的系统试试 就知道了， 另外，我们可以到\mmt\sdcard\gestures获取到保存手势的文件！ 好了，唠唠叨叨那么多，开始讲正题吧！

对了，贴下官方API文档先:GestureDetector

1.Android中手势交互的执行顺序

> - 1.手指触碰屏幕时，触发MotionEvent事件！
> - 2.该事件被OnTouchListener监听，可在它的onTouch()方法中获得该MotionEvent对象！
> - 3.通过GestureDetector转发MotionEvent对象给OnGestureListener
> - 4.我们可以通过OnGestureListener获得该对象，然后获取相关信息，以及做相关处理！

我们来看下上述的三个类都是干嘛的: **MotionEvent**: 这个类用于封装手势、触摸笔、轨迹球等等的动作事件。 其内部封装了两个重要的属性X和Y，这两个属性分别用于记录横轴和纵轴的坐标。 **GestureDetector**: 识别各种手势。 **OnGestureListener**: 这是一个手势交互的监听接口，其中提供了多个抽象方法，并根据GestureDetector的手势识别结果调用相对应的方法。

——上述资料摘自:http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2012/1020/448.html

2.GestureListener详解：

从1中我们知道了监听手势的关键是:GestureListener 他给我们提供了下述的回调方法：

> - 按下（onDown）：刚刚手指接触到触摸屏的那一刹那，就是触的那一下。
> - 抛掷（onFling）：手指在触摸屏上迅速移动，并松开的动作。
> - 长按（onLongPress）：手指按在持续一段时间，并且没有松开。
> - 滚动（onScroll）：手指在触摸屏上滑动。
> - 按住（onShowPress）：手指按在触摸屏上，它的时间范围在按下起效，在长按之前。
> - 抬起（onSingleTapUp）：手指离开触摸屏的那一刹那。

知道了GestureListener的相关方法后，实现手势检测也很简单，步骤如下：

> - Step 1: 创建GestureDetector对象，创建时需实现GestureListener传入
> - Step 2: 将Activity或者特定组件上的TouchEvent的事件交给GestureDetector处理即可！我们写个简单的代码来验证这个流程，即重写对应的方法：

代码如下：

```
public class MainActivity extends AppCompatActivity {

    private MyGestureListener mgListener;
    private GestureDetector mDetector;
    private final static String TAG = "MyGesture";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //实例化GestureListener与GestureDetector对象
        mgListener = new MyGestureListener();
        mDetector = new GestureDetector(this, mgListener);

    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return mDetector.onTouchEvent(event);
    }

    //自定义一个GestureListener,这个是View类下的，别写错哦！！！
    private class MyGestureListener implements GestureDetector.OnGe
```

```
        @Override
        public boolean onDown(MotionEvent motionEvent) {
            Log.d(TAG, "onDown:按下");
            return false;
        }

        @Override
        public void onShowPress(MotionEvent motionEvent) {
            Log.d(TAG, "onShowPress:手指按下一段时间,不过还没到长按");
        }

        @Override
        public boolean onSingleTapUp(MotionEvent motionEvent) {
            Log.d(TAG, "onSingleTapUp:手指离开屏幕的一瞬间");
            return false;
        }

        @Override
        public boolean onScroll(MotionEvent motionEvent, MotionEver
            Log.d(TAG, "onScroll:在触摸屏上滑动");
            return false;
        }

        @Override
        public void onLongPress(MotionEvent motionEvent) {
            Log.d(TAG, "onLongPress:长按并且没有松开");
        }

        @Override
        public boolean onFling(MotionEvent motionEvent, MotionEvent
            Log.d(TAG, "onFling:迅速滑动, 并松开");
            return false;
        }
    }

}
```

对应操作截图：

- 1.按下后立即松开:
  onDown:按下
  onSingleTapUp:手指离开屏幕的一瞬间
- 2.长按后松开：
  onDown:按下
  onShowPress:手指按下一段时间,不过还没到长按
  onLongPress:长按并且没有松开
- 3.轻轻一滑，同时松开：
  onDown:按下
  onScroll:在触摸屏上滑动
  onScroll:在触摸屏上滑动
  onFling:迅速滑动，并松开

- 4.按住后不放持续做滑动操作：

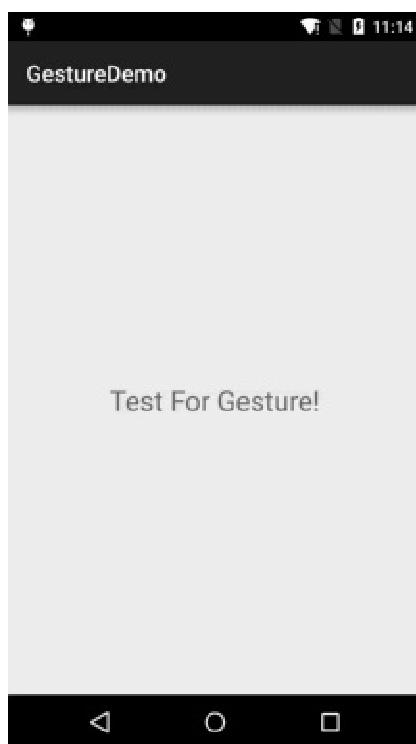
```
onDown:按下
onShowPress:手指按下一段时间,不过还没到长按
onScroll:在触摸屏上滑动
onScroll:在触摸屏上滑动
```

PS:从上述结果来看，我们发现了一个问题：我们实现OnGestureListener需要实现所有的手势，可能我针对的仅仅是滑动，但是你还是要去重载，这显得很逗逼，是吧，官方肯定会给出解决方法滴，官方另外给我们提供了一个SimpleOnGestureListener类 只需把上述的OnGestureListener替换成SimpleOnGestureListener即可！

3.简单的例子:下滑关闭Activity，上滑启动新的Activity

这里就用上述的SimpleOnGestureListener来实现吧:

运行效果图：



实现代码：

```
public class MainActivity extends AppCompatActivity {

    private GestureDetector mDetector;
    private final static int MIN_MOVE = 200;   //最小距离
    private MyGestureListener mgListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //实例化SimpleOnGestureListener与GestureDetector对象
        mgListener = new MyGestureListener();
        mDetector = new GestureDetector(this, mgListener);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return mDetector.onTouchEvent(event);
    }

    //自定义一个GestureListener,这个是View类下的，别写错哦！！！
    private class MyGestureListener extends GestureDetector.SimpleO

        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, floa
            if(e1.getY() - e2.getY() > MIN_MOVE){
                startActivity(new Intent(MainActivity.this, MainAct
                Toast.makeText(MainActivity.this, "通过手势启动Activi
            }else if(e1.getY() - e2.getY()  < MIN_MOVE){
                finish();
                Toast.makeText(MainActivity.this,"通过手势关闭Activit
            }
            return true;
        }
    }

}
```

结果分析： 从上面的对比就可以知道，相比起SimpleOnGestureListener使用SimpleOnGestureListener 显得更加的简单，想重写什么方法就重写什么方法， 另外例子比较简单，大家可以自己试试 其他玩法，比如通过手势缩放图片~

4.手势添加与识别：

> 除了上面讲解的手势检测外，Android还运行我们将手势进行添加，然后提供了相关的识别API；Android中使用GestureLibrary来代表手势库，提供了GestureLibraries工具类来创建手势库！

四个加载手势库的静态方法：

static GestureLibrary	fromFile (File path)
static GestureLibrary	fromFile (String path)
static GestureLibrary	fromPrivateFile (Context context, String name)
static GestureLibrary	fromRawResource (Context context, int resourceId)

获得GestureLibraries对象后，就可以使用该对象提供的下述方法来做相应操作了：

相关方法：

- public void **addGesture** (String entryName, Gesture gesture)：添加一个名为entryName的手势
- public Set<String> **getGestureEntries** ()：获得手势库中所有手势的名称
- public ArrayList<Gesture> **getGestures** (String entryName)：获得entryName名称对应的全部手势
- public ArrayList<Prediction> **recognize** (Gesture gesture): 从当前手势库中识别与gesture匹配的全部手势
- public void **removeEntry** (String entryName)：删除手势库中entryName名称对应的手势
- public void **removeGesture** (String entryName, Gesture gesture)：删除手势库中entryName和gesture都匹配的手势
- public abstract boolean **save** ()：想手势库中添加手势或从中删除手势后调用该方法保存手势库

GestureOverlayView手势编辑组件：

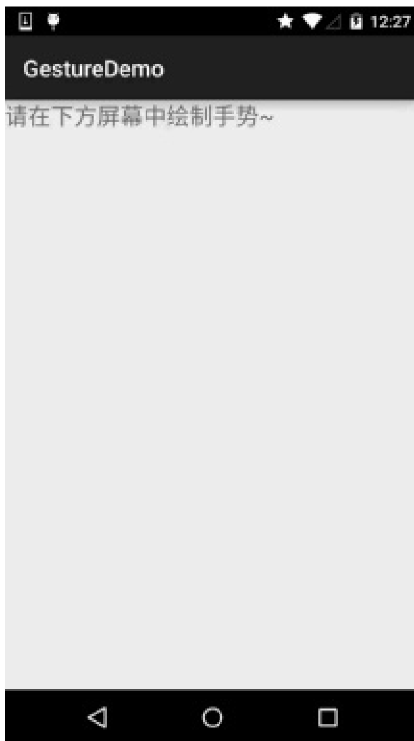
Android为GestureOverlayView提供了三种监听器接口，如下，一般常用的是:**OnGesturePerformedListener**;用于手势完成时提供响应！

interface	GestureOverlayView.OnGestureListener
interface	GestureOverlayView.OnGesturePerformedListener
interface	GestureOverlayView.OnGesturingListener

5.手势添加示例：

PS：例子引用的是——李刚《Android疯狂讲义》的代码

运行效果图：



好吧，下面贴下实现代码：

两个布局文件：activity_main.xml和dialog_save.xml

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="请在下方屏幕中绘制手势~"
        android:textSize="20sp"/>

    <!-- gestureStrokeType控制手势是否需要一笔完成,multiple表示允许多笔--
    <android.gesture.GestureOverlayView
        android:id="@+id/gesture"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gestureStrokeType="multiple" />

</LinearLayout>
```

dialog_save.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="8dp"
            android:text="请填写手势名称："/>
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/edit_name"/>
    </LinearLayout>

    <ImageView
        android:id="@+id/img_show"
        android:layout_width="128dp"
        android:layout_height="128dp"
        android:layout_marginTop="10dp"/>

</LinearLayout>
```

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    private EditText editText;
    private GestureOverlayView gesture;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获取手势编辑组件后，设置相关参数
        gesture = (GestureOverlayView) findViewById(R.id.gesture);
        gesture.setGestureColor(Color.GREEN);
        gesture.setGestureStrokeWidth(5);
        gesture.addOnGesturePerformedListener(new GestureOverlayVie
            @Override
            public void onGesturePerformed(GestureOverlayView gestu
                View saveDialog = getLayoutInflater().inflate(R.lay
                ImageView img_show = (ImageView) saveDialog.findVie
                final EditText edit_name = (EditText) saveDialog.fi
                Bitmap bitmap = gesture.toBitmap(128,128,10,0xffff(
                img_show.setImageBitmap(bitmap);
                new AlertDialog.Builder(MainActivity.this).setView(
                        .setPositiveButton("保存",new DialogInterfa
                        {
                            @Override
                            public void onClick(DialogInterface dia
                                //获取文件对应的手势库
                                GestureLibrary gestureLib = Gesture
                                gestureLib.addGesture(edit_name.get
                                gestureLib.save();
                            }
                        }).setNegativeButton("取消", null).show();
            }
        });
    }
}
```

最后还需要在AndroidManifest.xml中添加写入SD卡的权限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_S
```

6.手势识别示例

实现代码：

```
public class MainActivity extends AppCompatActivity {

    private GestureOverlayView gesture;
    private GestureLibrary gestureLibrary;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        gestureLibrary = GestureLibraries.fromFile("mmt/sdcard/myge
        if (gestureLibrary.load()) {
            Toast.makeText(mContext, "手势库加载成功", Toast.LENGTH_S
        } else {
            Toast.makeText(mContext, "手势库加载失败", Toast.LENGTH_S
        }

        //获取手势编辑组件后，设置相关参数
        gesture = (GestureOverlayView) findViewById(R.id.gesture);
        gesture.setGestureColor(Color.GREEN);
        gesture.setGestureStrokeWidth(5);
        gesture.addOnGesturePerformedListener(new GestureOverlayVie
            @Override
            public void onGesturePerformed(GestureOverlayView gestu
                //识别用户刚绘制的手势
                ArrayList<Prediction> predictions = gestureLibrary.
                ArrayList<String> result = new ArrayList<String>();
                //遍历所有找到的Prediction对象
                for (Prediction pred : predictions) {
                    if (pred.score > 2.0) {
                        result.add("与手势【" + pred.name + "】相似度
                    }
                }
                if (result.size() > 0) {
                    ArrayAdapter<Object> adapter = new ArrayAdapter
                            android.R.layout.simple_dropdown_item_1
                    new AlertDialog.Builder(mContext).setAdapter(ad
                }else{
                    Toast.makeText(mContext,"无法找到匹配的手势！",Toa
                }
            }
        });
    }
}
```

另外别忘了在AndroidManifest.xml文件中加入读SD卡的权限：

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STO
```

本节小结：

好的，本节介绍了Android中的Gesture手势，讲解了手势判断，手势添加，手势识别三个内容，大部分例子来自于李刚老师的Android疯狂讲义，有兴趣的可以看看该书~谢谢

4.1.1 Activity初学乍练

本节引言：

本节开始讲解Android的四大组件之一的Activity(活动)，先来看下官方对于Activity的介绍： PS:官网文档：Activity

介绍如下：

> An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows.

大概意思：

> Activity是一个应用程序的组件，他在屏幕上提供了一个区域，允许用户在上面做一些交互性的操作， 比如打电话，照相，发送邮件，或者显示一个地图！ Activity可以理解成一个绘制用户界面的窗口， 而这个窗口可以填满整个屏幕，也可能比屏幕小或者浮动在其他窗口的上方！

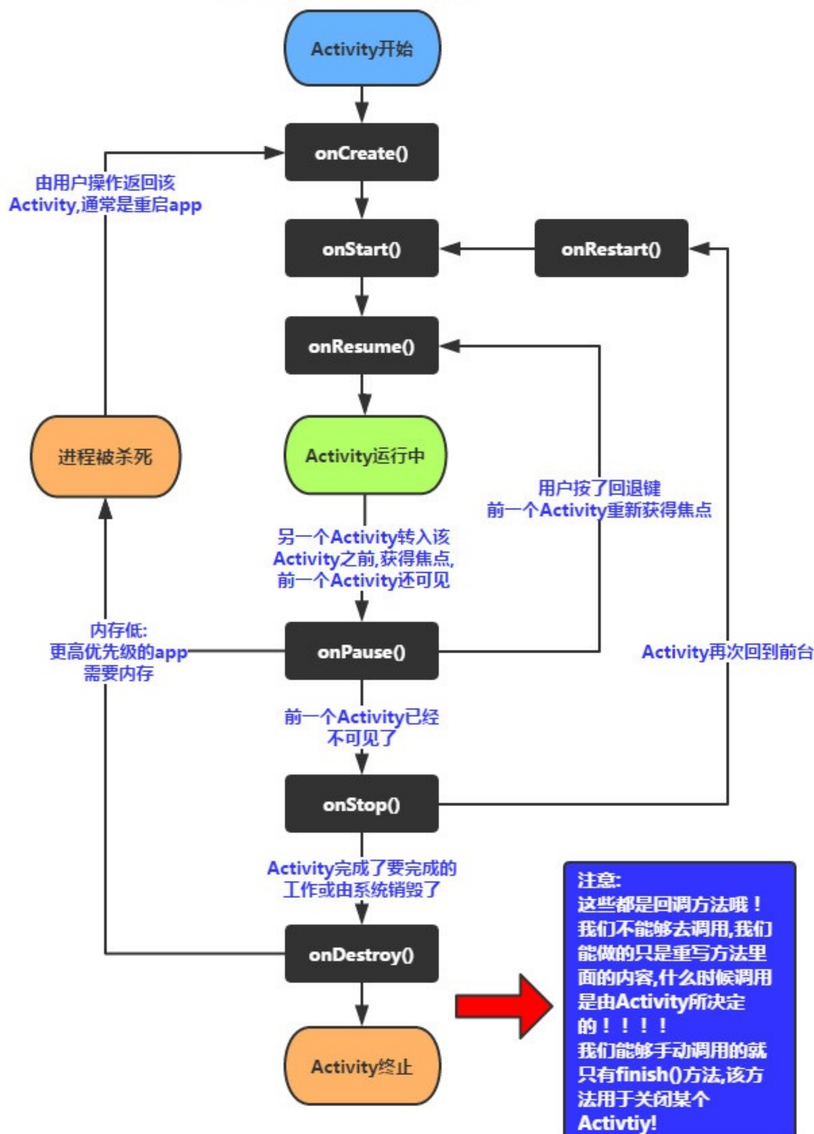
从上面这段话，我们可以得到以下信息：

> **1.** Activity用于显示用户界面，用户通过Activity交互完成相关操作 **2.** 一个App允许有多个Activity

好了，大概的引言就介绍到这里，想深入了解可以继续看API，开始本节内容~

1.Activity的概念与Activity的生命周期图：





注意事项：

> **1. onPause()和onStop()**被调用的前提是： 打开了一个新的Activity！而前者是旧Activity还可见的状态；后者是旧Activity已经不可见！ 2. 另外， 亲测：AlertDialog和PopWindow是不会触发上述两个回调方法的~

2.Activity/ActionBarActivity/AppCompatActivity的区别：

> 在开始讲解创建Activity之前要说下这三个的一个区别： Activity就不用说啦，后面这两个都是为了低版本兼容而提出的提出来的，他们都在v7包下，ActionBarActivity已被废弃，从名字就知道，ActionBar~，而在5.0后，被Google弃用了，现在用 ToolBar...而我们现在在Android Studio创建一个Activity默认继承的会是：AppCompatActivity! 当然你也可以只写Activity，不过AppCompatActivity给我们提供了一些新的东西而已！ 两个选一个，Just you like~

3.Activity的创建流程

Activity的使用流程

①自定义Activity类名,继承Activity类或者它的子类

```
class MyActivity extends Activity{
```

②重写onCreate()方法,在该方法中调setContentView()设置要显示的视图

```
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
}
```

③在AndroidManifest.xml对Activity进行配置

```
<activity
      android:icon = "图标"
      android:name = "类名"
      android:label = "Activity显示的标题"
      android:theme = "要应用的主题"></activity>
```

④启动Activity:调用startActivity(Intent);

```
Intent it = new
Intent(MainActivity.this,MyActivity.class) ;
startActivity(it);
```

⑤关闭Activity:调用finish,直接关闭当前Activity

我们可以把他写到启动第二个Activity的方法中,当启动第二个Activity时,第一个Activity就会被关闭

```
finish() ;
```

PS:

> 好了，上面也说过，可以继承Activity和AppCompatActivity，只不过后者提供了一些新的东西而已！ 另外，切记，Android中的四大组件，只要你定义了，无论你用没用，都要在AndroidManifest.xml对 这个组件进行声明，不然运行时程序会直接退出，报ClassNotFindException...

4.onCreate()一个参数和两个参数的区别：

相信用as的朋友在重写Act的onCreate()方法时会发现，这玩意有两个参数：

```
@Override
public void onCreate(Bundle savedInstanceState, PersistableBundle persistentState) {
    super.onCreate(savedInstanceState, persistentState);
}
```

可是正常的才只有一个参数啊：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

恩呢，这就是5.0给我们提供的新的方法，要用它，先要在配置文件中为我们的Activity设置一个属性：

```
android:persistableMode="persistAcrossReboots"
```

然后我们的Activity就拥有了持久化的能力了，一般我们会搭配另外两个方法来使用：

```
public void onSaveInstanceState(Bundle outState, PersistableBundle
public void onRestoreInstanceState(Bundle savedInstanceState, Pers:
```

相信有些朋友对这两个方法名不陌生吧，前一个方法会在下述情形中被调用：

> 1. 点击home键回到主页或长按后选择运行其他程序
> 2. 按下电源键关闭屏幕
> 3. 启动新的Activity
> 4. 横竖屏切换时，肯定会执行，因为横竖屏切换的时候会先销毁Act，然后再重新创建 重要原则：当系统"未经你许可"时销毁了你的activity，则onSaveInstanceState会被系统调用， 这是系统的责任，因为它必须要提供一个机会让你保存你的数据（你可以保存也可以不保存）。

而后一个方法，和onCreate同样可以从取出前者保存的数据： 一般是在onStart()和onResume()之间执行！ 之所以有两个可以获取到保存数据的方法，是为了避免Act跳转而没有关闭， 然后不走onCreate()方法，而你又想取出保存数据~

说回来： 说回这个Activity拥有了持久化的能力，增加的这个PersistableBundle参数令这些方法 拥有了系统关机后重启的数据恢复能力！！而且不影响我们其他的序列化操作，卧槽， 具体怎么实现的，暂时还不了解，可能是另外弄了个文件保存吧~！后面知道原理的话会告知下大家！ 另外，API版本需要>=21，就是要5.0以上的版本才有效~

4.启动一个Activity的几种方式

> 在Android中我们可以通过下面两种方式来启动一个新的Activity,注意这里是怎么启动，而非 启动模式！！分为显示启动和隐式启动！

1. 显式启动：通过包名来启动，写法如下：

> ①最常见的：
>
> ```
> startActivity(new Intent(当前Act.this,要启动的Act.class));
> ```
>
> ②通过**Intent**的**ComponentName**：
>
> ```
> ComponentName cn = new ComponentName("当前Act的全限定类名","启动Ac
> Intent intent = new Intent() ;
> intent.setComponent(cn) ;
> startActivity(intent) ;
> ```
>
> ③初始化**Intent**时指定包名：
>
> ```
> Intent intent = new Intent("android.intent.action.MAIN");
> intent.setClassName("当前Act的全限定类名","启动Act的全限定类名");
> startActivity(intent);
> ```

2.隐式启动：通过**Intent-filter**的**Action,Category**或**data**来实现 这个是通过**Intent** 的 intent-filter**来实现的，这个Intent那章会详细讲解！ 这里知道个大概就可以了！

```
<activity android:name=".SecondActivity"
    android:label="第二个Activity">
    <intent-filter>
        <action android:name="my_action"/>
        <category android:name="my_category"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

！这个一定要写哦~

Java文件中启动:

```
Intent it = new Intent();
it.setAction("my_action");
it.addCategory("my_category");
startActivity(it);
```

3. 另外还有一个直接通过包名启动**apk**的：

```
Intent intent = getPackageManager().getLaunchIntentForPackage
("apk第一个启动的Activity的全限定类名") ;
if(intent != null) startActivity(intent) ;
```

5.横竖屏切换与状态保存的问题

> 前面也也说到了App横竖屏切换的时候会销毁当前的Activity然后重新创建一个，你可以自行在生命周期 的每个方法里都添加打印Log的语句，来进行判断，又或者设一个按钮一个TextView点击按钮后，修改TextView 文本，然后横竖屏切换，会神奇的发现TextView文本变回之前的内容了！ 横竖屏切换时Act走下述生命周期： **onPause-> onStop-> onDestory-> onCreate->onStart->onResume** 关于横竖屏切换可能遇到下述问题：

1.先说下如何禁止屏幕横竖屏自动切换吧，很简单，在AndroidManifest.xml中为Act添加一个属性： **android:screenOrientation**， 有下述可选值：

- **unspecified**:默认值 由系统来判断显示方向.判定的策略是和设备相关的，所以不同的设备会有不同的显示方向。
- **landscape**:横屏显示（宽比高要长）
- **portrait**:竖屏显示(高比宽要长)
- **user**:用户当前首选的方向
- **behind**:和该Activity下面的那个Activity的方向一致(在Activity堆栈中的)
- **sensor**:有物理的感应器来决定。如果用户旋转设备这屏幕会横竖屏切换。
- **nosensor**:忽略物理感应器，这样就不会随着用户旋转设备而更改了（"unspecified"设置除外）。

2.横竖屏时想加载不同的布局：

1）准备两套不同的布局，Android会自己根据横竖屏加载不同布局： 创建两个布局文件夹：**layout-land**横屏,**layout-port**竖屏 然后把这两套布局文件丢这两文件夹里，文件名一样，Android就会自行判断，然后加载相应布局了！

2)自己在代码中进行判断，自己想加载什么就加载什么：

我们一般是在onCreate()方法中加载布局文件的，我们可以在这里对横竖屏的状态做下判断，关键代码如下：

```
if (this.getResources().getConfiguration().orientation == Configura
     setContentView(R.layout.横屏);
}

else if (this.getResources().getConfiguration().orientation ==Confi
    setContentView(R.layout.竖屏);
}
```

3. 如何让模拟器横竖屏切换

如果你的模拟器是GM的话。直接按模拟器上的切换按钮即可，原生模拟器可按ctrl + f11/f12切换！

4. 状态保存问题：

这个上面也说过了，通过一个Bundle savedInstanceState参数即可完成！ 三个核心方法：

```
onCreate(Bundle savedInstanceState);
onSaveInstanceState(Bundle outState);
onRestoreInstanceState(Bundle savedInstanceState);
```

你只重写onSaveInstanceState()方法，往这个bundle中写入数据，比如：

> outState.putInt("num",1);

这样，然后你在onCreate或者onRestoreInstanceState中就可以拿出里面存储的数据，不过拿之前要判断下是否为null哦！

> savedInstanceState.getInt("num");

然后想干嘛就干嘛~

6.系统给我们提供的常见的Activity

好的，最后给大家附上一些系统给我们提供的一些常见的Activtiy吧！

```
//1.拨打电话
// 给移动客服10086拨打电话
Uri uri = Uri.parse("tel:10086");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);
```

```
//2.发送短信
// 给10086发送内容为“Hello”的短信
Uri uri = Uri.parse("smsto:10086");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
intent.putExtra("sms_body", "Hello");
startActivity(intent);

//3.发送彩信（相当于发送带附件的短信）
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra("sms_body", "Hello");
Uri uri = Uri.parse("content://media/external/images/media/23");
intent.putExtra(Intent.EXTRA_STREAM, uri);
intent.setType("image/png");
startActivity(intent);

//4.打开浏览器:
// 打开Google主页
Uri uri = Uri.parse("http://www.baidu.com");
Intent intent  = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//5.发送电子邮件:(阉割了Google服务的没戏!!!!)
// 给someone@domain.com发邮件
Uri uri = Uri.parse("mailto:someone@domain.com");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
startActivity(intent);
// 给someone@domain.com发邮件发送内容为“Hello”的邮件
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, "someone@domain.com");
intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
intent.setType("text/plain");
startActivity(intent);
// 给多人发邮件
Intent intent=new Intent(Intent.ACTION_SEND);
String[] tos = {"1@abc.com", "2@abc.com"}; // 收件人
String[] ccs = {"3@abc.com", "4@abc.com"}; // 抄送
String[] bccs = {"5@abc.com", "6@abc.com"}; // 密送
intent.putExtra(Intent.EXTRA_EMAIL, tos);
intent.putExtra(Intent.EXTRA_CC, ccs);
intent.putExtra(Intent.EXTRA_BCC, bccs);
intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
intent.setType("message/rfc822");
startActivity(intent);

//6.显示地图:
// 打开Google地图中国北京位置（北纬39.9，东经116.3）
Uri uri = Uri.parse("geo:39.9,116.3");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

```
//7.路径规划
// 路径规划：从北京某地（北纬39.9，东经116.3）到上海某地（北纬31.2，东经121.
Uri uri = Uri.parse("http://maps.google.com/maps?f=d&saddr=39.9 116
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//8.多媒体播放:
Intent intent = new Intent(Intent.ACTION_VIEW);
Uri uri = Uri.parse("file:///sdcard/foo.mp3");
intent.setDataAndType(uri, "audio/mp3");
startActivity(intent);

//获取SD卡下所有音频文件,然后播放第一首=-=
Uri uri = Uri.withAppendedPath(MediaStore.Audio.Media.INTERNAL_CONT
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//9.打开摄像头拍照:
// 打开拍照程序
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, 0);
// 取出照片数据
Bundle extras = intent.getExtras();
Bitmap bitmap = (Bitmap) extras.get("data");

//另一种:
//调用系统相机应用程序，并存储拍下来的照片
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
time = Calendar.getInstance().getTimeInMillis();
intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(new File(Envi
.getExternalStorageDirectory().getAbsolutePath()+"/tucue", time + '
startActivityForResult(intent, ACTIVITY_GET_CAMERA_IMAGE);

//10.获取并剪切图片
// 获取并剪切图片
Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
intent.setType("image/*");
intent.putExtra("crop", "true"); // 开启剪切
intent.putExtra("aspectX", 1); // 剪切的宽高比为1：2
intent.putExtra("aspectY", 2);
intent.putExtra("outputX", 20); // 保存图片的宽和高
intent.putExtra("outputY", 40);
intent.putExtra("output", Uri.fromFile(new File("/mnt/sdcard/temp")
intent.putExtra("outputFormat", "JPEG");// 返回格式
startActivityForResult(intent, 0);
// 剪切特定图片
Intent intent = new Intent("com.android.camera.action.CROP");
intent.setClassName("com.android.camera", "com.android.camera.CropI
intent.setData(Uri.fromFile(new File("/mnt/sdcard/temp")));
intent.putExtra("outputX", 1); // 剪切的宽高比为1：2
intent.putExtra("outputY", 2);
intent.putExtra("aspectX", 20); // 保存图片的宽和高
intent.putExtra("aspectY", 40);
```

```
intent.putExtra("scale", true);
intent.putExtra("noFaceDetection", true);
intent.putExtra("output", Uri.parse("file:///mnt/sdcard/temp"));
startActivityForResult(intent, 0);

//11.打开Google Market
// 打开Google Market直接进入该程序的详细页面
Uri uri = Uri.parse("market://details?id=" + "com.demo.app");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//12.进入手机设置界面:
// 进入无线网络设置界面（其它可以举一反三）
Intent intent = new Intent(android.provider.Settings.ACTION_WIRELES
startActivityForResult(intent, 0);

//13.安装apk:
Uri installUri = Uri.fromParts("package", "xxx", null);
returnIt = new Intent(Intent.ACTION_PACKAGE_ADDED, installUri);

//14.卸载apk:
Uri uri = Uri.fromParts("package", strPackageName, null);
Intent it = new Intent(Intent.ACTION_DELETE, uri);
startActivity(it);

//15.发送附件:
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.putExtra(Intent.EXTRA_STREAM, "file:///sdcard/eoe.mp3");
sendIntent.setType("audio/mp3");
startActivity(Intent.createChooser(it, "Choose Email Client"));

//16.进入联系人页面:
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setData(People.CONTENT_URI);
startActivity(intent);

//17.查看指定联系人:
Uri personUri = ContentUris.withAppendedId(People.CONTENT_URI, info
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setData(personUri);
startActivity(intent);
```

本节小结：

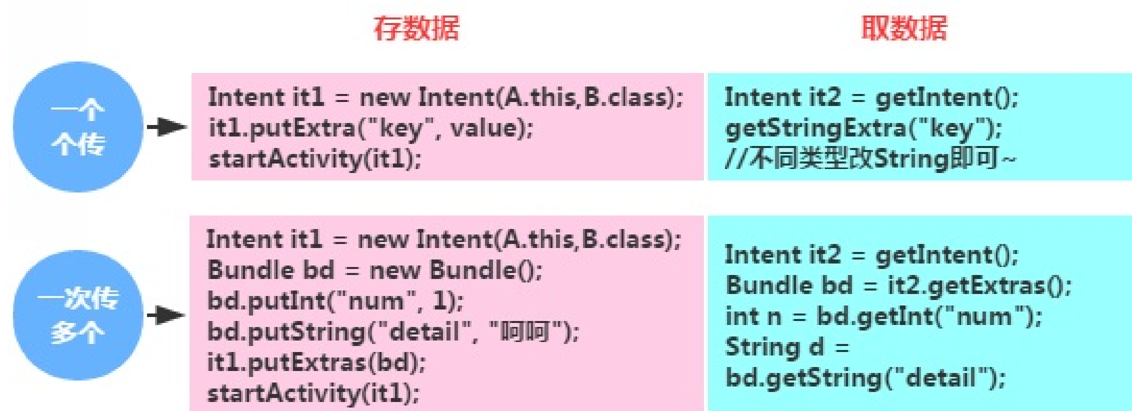
> 好吧，写着写着就不像入门教程了，哈哈，不过学多点没事的，本节初窥门径就到这里吧~下节我们会继续来研究这个Activity，比如数据传递，启动模式等~敬请期待~

4.1.2 Activity初窥门径

本节引言：

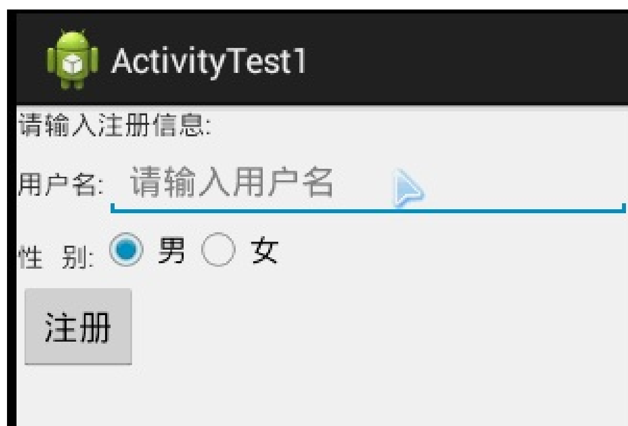
上一节中我们对Activity一些基本的概念进行了了解，什么是Activity，Activity的生命周期，如何去启动一个Activity等，本节我们继续来学习Activity，前面也讲了一个App一般都是又多个Activity构成的，这就涉及到了多个Activity间数据传递的问题了，那么本节继续学习Activity的使用！另外关于传递集合，对象，数组，Bitmap的我们会在Intent那里进行讲解，这里只介绍如何传递基本数据！

1.Activity间的数据传递：



代码示例：

效果图：



代码下载：ActivityTest1.zip

2.多个Activity间的交互(后一个传回给前一个)

①使用startActivityForResult(Intent intent, int requestCode)
来启动一个Activity
②在启动的Activity中重写
onActivityResult(int requestCode, int resultCode, Intent data)
requestCode是用于区分在该Activity中不同的启动方式,比如有两个不同的按钮,启动的是同一个Activity,传递的数据可能不同,这里就可以用这个requestCode来区别
resultCode:子Activity通过setResult()返回的码
③在子Activity重写:
setResult(int resultCode, Intent data)

代码示例：

效果图：



代码下载：ActivityTest2.zip

3.知晓当前是哪个Activity

让所有Activity继承一个自定义的BaseActivity类,在OnCreate()方法中添加下述语句即可:
Log.d("BaseActivity", getClass().getSimpleName());

4.随时关闭所有Activity

> 有时我们可能会打开了很多个Activity，突然来个这样的需求，在某个页面可以关掉 所有的Activity并退出程序！好吧，下面提供一个关闭所有Activity的方法， 就是用一个list集合来存储所有Activity!

1.创建一个Activity管理器类:ActivityCollector
定义三个共有静态的方法:定义存储Activity的list集合,方法如下
①addActivity:往集合添加Activity对象
②removeActivity:移除Activity中的对象
③finishAll:增强for循环遍历所有Activity调用:if(!activity.isFinishing( ))activity.finish();

2.BaseActivity中
①onCreate( )方法添加
ActivityCollector.addActivity(this);
②onDestory( )方法添加:
ActivityCollector.removeActivity(this);
③可以在任意一个Activity中调用:
ActivityCollector.finishAll( );
从而关闭所有Activity,退出app!

具体代码如下：

```
public class ActivityCollector {
    public static LinkedList<Activity> activities = new LinkedList<
    public static void addActivity(Activity activity)
    {
        activities.add(activity);
    }
    public static void removeActivity(Activity activity)
    {
        activities.remove(activity);
    }
    public static void finishAll()
    {
        for(Activity activity:activities)
        {
            if(!activity.isFinishing())
            {
                activity.finish();
            }
        }
    }
}
```

5.完全退出App的方法

上面说的是关闭所有Activity的，但是有些时候我们可能想杀死整个App，连后台任务都杀死 杀得一干二净的话，可以使用搭配着下述代码使用：

实现代码：

```
/**
 * 退出应用程序
 */
public void AppExit(Context context) {
    try {
        ActivityCollector.finishAll();
        ActivityManager activityMgr = (ActivityManager) context
                .getSystemService(Context.ACTIVITY_SERVICE);
        activityMgr.killBackgroundProcesses(context.getPackageName(
        System.exit(0);
    } catch (Exception ignored) {}
}
```

6.双击退出程序的两种方法：

1）定义一个变量，来标识是否退出

```
// 定义一个变量，来标识是否退出
private static boolean isExit = false;
Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        isExit = false;
    }
};

public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        if (!isExit) {
            isExit = true;
            Toast.makeText(getApplicationContext(), "再按一次退出程序
                    Toast.LENGTH_SHORT).show();
            // 利用handler延迟发送更改状态信息
            mHandler.sendEmptyMessageDelayed(0, 2000);
        } else {
            exit(this);
        }
        return false;
    }
return super.onKeyDown(keyCode, event);}
```

2）保存点击时间：

```
//保存点击的时间
private long exitTime = 0;
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        if ((System.currentTimeMillis() - exitTime) > 2000) {
            Toast.makeText(getApplicationContext(), "再按一次退出程序
                    Toast.LENGTH_SHORT).show();
            exitTime = System.currentTimeMillis();
        } else {
                        exit();
                      }
        return false;
    }
        return super.onKeyDown(keyCode, event);
}
```

7.为Activity设置过场动画

> 所谓的过场动画就是切换到另外的Activity时加上一些切换动画，比如淡入淡出，放大缩小，左右互推等！ 当然，我们并不在这里详细讲解动画，后面有专门的章节来讲解这个，这里只教大家如何去加载动画，另外 给大家提供了一些比较常用的过渡动画，只要将相关动画文件添加到res/anim目录下，然后下述方法二选一 就可以实现Activity的切换动画了！

1）方法一：

A跳转到B,在startActivity（intent）后面加上
overridePendingTransition(R.anim.anim_in, R.anim.anim_out);

B返回A,要在finish()后面加上
overridePendingTransition(R.anim.anim_in, R.anim.anim_out);

anim_in是进入的Activity的动画
anim_out是退出的Activity的动画

2）方法二：

通过style进行配置，这个是全局的哦，就是所有的Activity都会加载这个动画！

实现代码如下：

①在style.xml中自定义style：

```
<!-- 默认Activity跳转动画 -->
<style name="default_animation" mce_bogus="1" parent="@android:styl
    <item name="android:activityOpenEnterAnimation">@anim/default_a
    <item name="android:activityOpenExitAnimation">@anim/anim_stay<
    <item name="android:activityCloseEnterAnimation">@anim/anim_sta
    <item name="android:activityCloseExitAnimation">@anim/default_a
</style>
```

解释：

4个item分别代表:

- Activity A跳转到Activity B时Activity B进入动画;
- Activity A跳转到Activity B时Activity A退出动画;
- Activity B返回Activity A时Activity A的进入动画
- Activity B返回Activity A时ActivityB的退出动画

②然后修改下AppTheme:

```
<style name="AppTheme" mce_bogus="1" parent="@android:style/Theme.L
        <item name="android:windowAnimationStyle">@style/default_an
        <item name="android:windowNoTitle">true</item>
</style>
```

③最后在appliction设置下：

```
<application
   android:icon="@drawable/logo"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >
```

好的，动画特效就这样duang一声设置好了~

3）其他

好的，除了上面两种方法以外，还可以使用**TransitionManager**来实现，但是需求版本是API 19以上的， 另外还有一种**addOnPreDrawListener**的转换动画，这个用起来还是有点麻烦的，可能不是适合初学者 这里也不讲，最后提供下一些常用的动画效果打包，选择需要的特效加入工程即可！ Activity常用过渡动画.zip

8.Bundle传递数据的限制

> 在使用Bundle传递数据时，要注意，Bundle的大小是有限制的 < 0.5MB，如果大于这个值 是会报TransactionTooLargeException异常的！！！

9.使用命令行查看当前所有Activity的命令：

> 使用下述命令即可，前提是你为SDK配置了环境变量:**adb shell dumpsys activity**

10.设置Activity全屏的方法：

1）代码隐藏ActionBar

在Activity的onCreate方法中调用getActionBar.hide();即可

2）通过requestWindowFeature设置

requestWindowFeature(Window.FEATURE_NO_TITLE); 该代码需要在setContentView ()之前调用，不然会报错！！！

> 注： 把 requestWindowFeature(Window.FEATURE_NO_TITLE);放在super.onCreate(savedInstanceState);前面就可以隐藏ActionBar而不报错。

3）通过AndroidManifest.xml的theme

在需要全屏的Activity的标签内设置 theme = @android:style/Theme.NoTitleBar.FullScreen

11.onWindowFocusChanged方法妙用：

我们先来看下官方对这个方法的介绍：

abstract void	onWindowFocusChanged (boolean hasFocus) Callback method to be invoked when the window focus changes in the view tree.

就是，当Activity得到或者失去焦点的时候，就会回调该方法！ 如果我们想监控Activity是否加载完毕，就可以用到这个方法了~ 想深入了解的可移步到这篇文章：onWindowFocusChanged触发简介

12.定义对话框风格的Activity

> 在某些情况下，我们可能需要将Activity设置成对话框风格的，Activity一般是占满全屏的， 而Dialog则是占据部分屏幕的！实现起来也很简单！

直接设置下Activity的theme:

```
android:theme="@android:style/Theme.Dialog"
```

这样就可以了，当然你可以再设置下标题，小图标！

```
//设置左上角小图标
requestWindowFeature(Window.FEATURE_LEFT_ICON);
setContentView(R.layout.main);
getWindow().setFeatureDrawableResource(Window.FEATURE_LEFT_ICON, ar
//设置文字:
setTitle(R.string.actdialog_title);  //XML代码中设置:android:label="
```

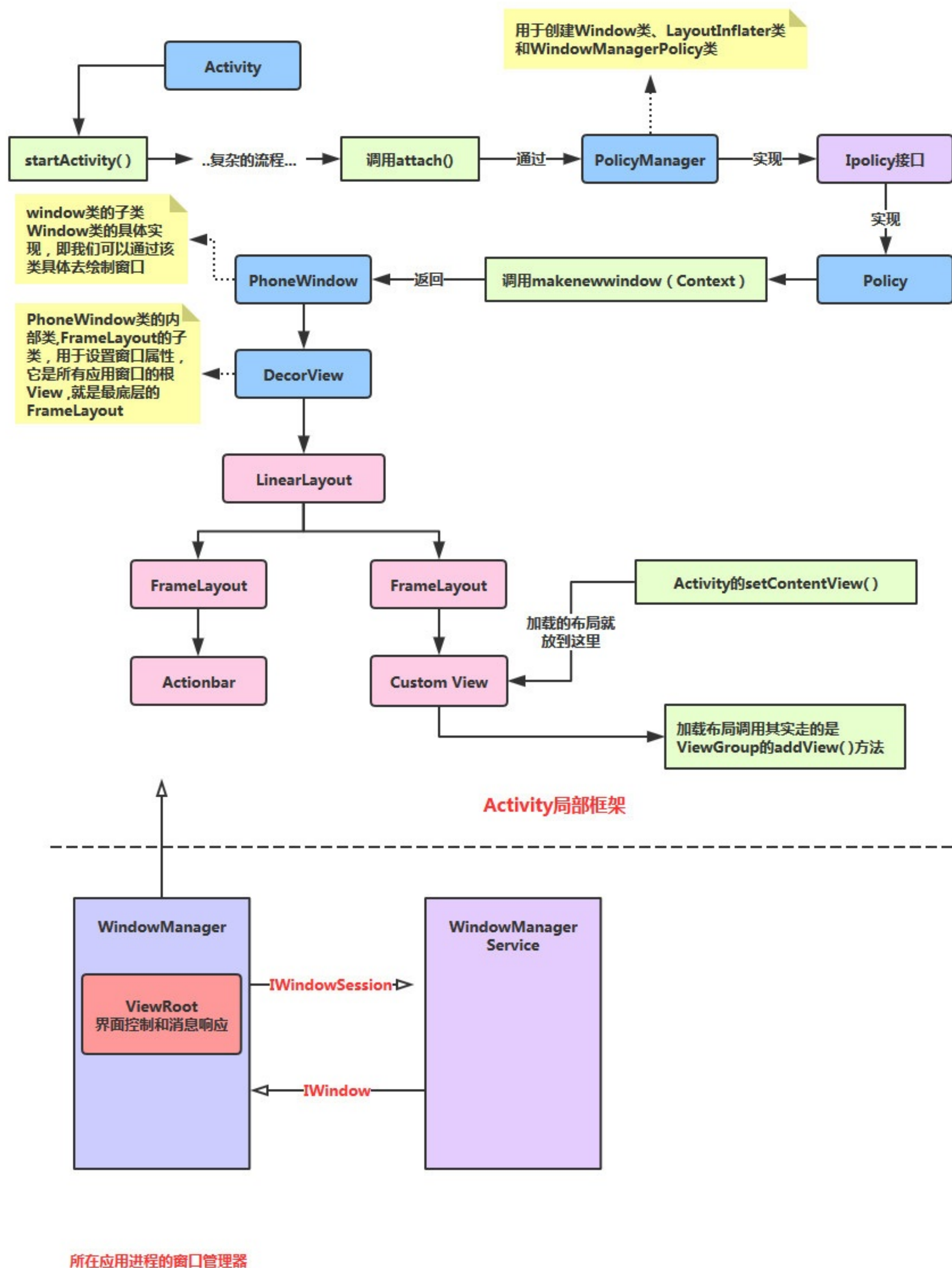
本节小结：

好的，本节我们又学习了一下Activity在实际开发中的一些常见问题，相信在实际开发中会帮到大家的！ 下节我们来学习Activty的栈的概念，以及四种加载模式！敬请期待~谢谢~

4.1.3 Activity登堂入室

1.Activity，Window与View的关系

> 好吧，本来就想了解下他们几个的关系，然后手多多，然后就开始看起他们的调用过程来了...结果扣了两个小时，只理解了很小很小的一部分，果然，到底层撸源码的都是大神，比如老罗，还没到那个等级，下面是自己查阅资料，看了下一点源码的归纳所得，如果哪写错了欢迎指出！下面贴下小结图：



流程解析： Activity调用startActivity后最后会调用attach方法，然后在PolicyManager实现一个Ipolicy接口，接着实现一个Policy对象，接着调用makenewwindow(Context)方法，该方法会返回一个PhoneWindow对象，而PhoneWindow 是Window的子类，在这个PhoneWindow中有一个DecorView的内部类，是所有应用窗口的根View，即View的老大， 直接控制Activity是否显示(引用

老司机原话..)，好吧，接着里面有一个LinearLayout，里面又有两个FrameLayout他们分别拿来装ActionBar和CustomView，而我们setContentView()加载的布局就放到这个CustomView中！

总结下这三者的关系：打个牵强的比喻：我们可以把这三个类分别堪称：画家，画布，画笔画出的东西；画家通过画笔(**LayoutInflater.infalte**)画出图案，再绘制在画布(**addView**)上！最后显示出来(**setContentView**)

2.Activity，Task和Back Stack的一些概念

接着我们来了解Android中Activity的管理机制，这就涉及到了两个名词：Task和Back Stack了！

概念解析：

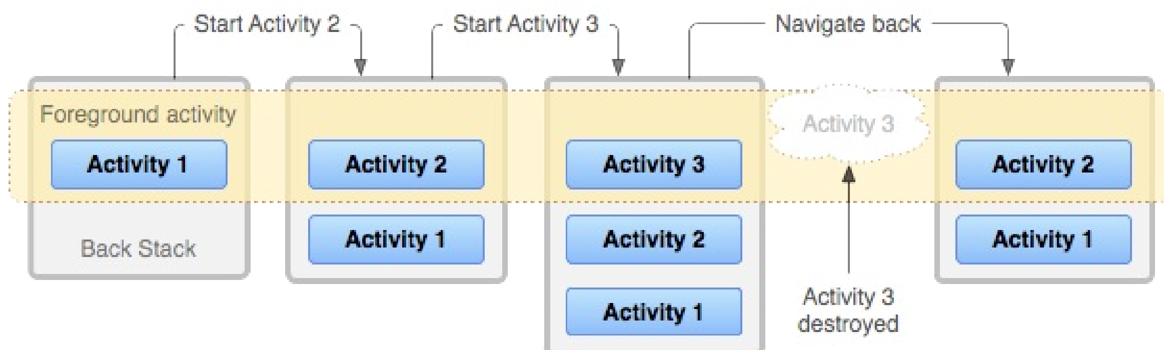
我们的APP一般都是由多个Activity构成的，而在Android中给我们提供了一个**Task(任务)**的概念，就是将多个相关的Activity收集起来，然后进行Activity的跳转与返回！当然，这个Task只是一个 frameworker层的概念，而在Android中实现了Task的数据结构就是**Back Stack**（回退堆栈）！相信大家对于栈这种数据结构并不陌生，Java中也有个Stack的集合类！栈具有如下特点：

> 先进先出(**LIFO**)，常用操作入栈(**push**)，出栈(**pop**)，处于最顶部的叫栈顶，最底部叫栈底

而Android中的Stack Stack也具有上述特点，他是这样来管理Activity的：

> 当切换到新的Activity，那么该Activity会被压入栈中，成为栈顶！而当用户点击Back键，栈顶的Activity出栈，紧随其后的Activity来到栈顶！

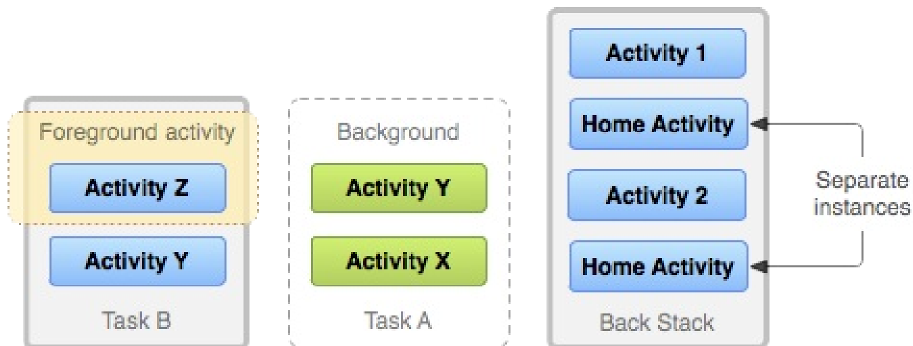
我们来看下官方文档给出的一个流程图：



流程解析：

应用程序中存在A1,A2,A3三个activity，当用户在Launcher或Home Screen点击应用程序图标时，启动主A1，接着A1开启A2，A2开启A3，这时栈中有三个Activity，并且这三个Activity默认在 同一个任务（Task）中，当用户按返回时，弹出A3，栈中只剩A1和A2，再按返回键，弹出A2，栈中只剩A1，再继续按返回键，弹出A1，任务被移除，即程序退出！

接着在官方文档中又看到了另外两个图，处于好奇，我又看了下解释，然后跟群里的人讨论了下：



然后还有这段解释：

A task is a cohesive unit that can move to the "background" when users begin a new task or go to the Home screen, via the *Home* button. While in the background, all the activities in the task are stopped, but the back stack for the task remains intact—the task has simply lost focus while another task takes place, as shown in figure 2. A task can then return to the "foreground" so users can pick up where they left off. Suppose, for example, that the current task (Task A) has three activities in its stack—two under the current activity. The user presses the *Home* button, then starts a new application from the application launcher. When the Home screen appears, Task A goes into the background. When the new application starts, the system starts a task for that application (Task B) with its own stack of activities. After interacting with that application, the user returns Home again and selects the application that originally started Task A. Now, Task A comes to the foreground—all three activities in its stack are intact and the activity at the top of the stack resumes. At this point, the user can also switch back to Task B by going Home and selecting the application icon that started that task (or by selecting the app's task from the overview screen). This is an example of multitasking on Android.

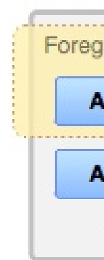


Figure 2. T
interactio
the backg

然后总结下了结论：

> Task是Activity的集合，是一个概念，实际使用的Back Stack来存储Activity，可以有多个Task，但是 同一时刻只有一个栈在最前面，其他的都在后台！那栈是如何产生的呢？
>
> 答：当我们通过主屏幕，点击图标打开一个新的App，此时会创建一个新的Task！举个例子： 我们通过点击通信录APP的图标打开APP，这个时候会新建一个栈1，然后开始把新产生的Activity添加进来，可能我们在通讯录的APP中打开了短信APP的页面，但是此时不会新建一个栈，而是继续添加到栈1中，这是 Android推崇一种用户体验方式，即不同应用程序之间的切换能使用户感觉就像是同一个应用程序， 很连贯的用户体验，官方称其为seamless (无缝衔接）！ ——————这个时候假如我们点击Home键，回到主屏幕，此时栈1进入后台，我们可能有下述两种操作： 1）点击菜单键(正方形那个按钮)，点击打开刚刚的程序，然后栈1又回到前台了！ 又或者我们点击主屏幕上通信录的图标，打开APP，此时也不会创建新的栈，栈1回到前台！ 2）如果此时我们点击另一个图标打开一个新的APP，那么此时则会创建一个新的栈2，栈2就会到前台， 而栈1继续呆在后台； 3) 后面也是这样...以此类推！

3.Task的管理

1）文档翻译：

好的，继续走文档，从文档中的ManagingTasks开始，大概的翻译如下：

1）文档翻译

继续走文档，从文档中的ManagingTasks开始，翻译如下：

> 如上面所述，Android会将新成功启动的Activity添加到同一个Task中并且按照以"先进先出"方式管理多个Task 和Back Stack，用户就无需去担心Activites如何与Task任务进行交互又或者它们是如何存在于Back Stack中！ 或许，你想改变这种正常的管理方式。比如，你希望你的某个Activity能够在一个新的Task中进行管理； 或者你只想对某个Activity进行实例化，又或者你想在用户离开任务时清理Task中除了根Activity所有Activities。你可以做这些事或者更多，只需要通过修改AndroidManifest.xml中 < activity >的相关属性值或者在代码中通过传递特殊标识的Intent给startActivity( )就可以轻松的实现 对Actvitiy的管理了。

< activity >中我们可以使用的属性如下：

> - **taskAffinity**
> - **launchMode**
> - **allowTaskReparenting**
> - **clearTaskOnLaunch**
> - **alwaysRetainTaskState**
> - **finishOnTaskLaunch**

你能用的主要的**Intent**标志有：

> - **FLAG_ACTIVITY_NEW_TASK**
> - **FLAG_ACTIVITY_CLEAR_TOP**
> - **FLAG_ACTIVITY_SINGLE_TOP**

好的，接下来逐个介绍这些怎么用：

2）taskAffinity和allowTaskReparenting

默认情况下，一个应用程序中的所有**activity**都有一个**Affinity**，这让它们属于同一个Task。 你可以理解为是否处于同一个Task的标志，然而，每个Activity可以通过< activity>中的taskAffinity属性设置单独的Affinity。 不同应用程序中的Activity可以共享同一个Affinity，同一个应用程序中的不同Activity 也可以设置成不同的Affinity。 Affinity属性在2种情况下起作用：

> 1）当启动 activity的Intent对象包含**FLAG_ACTIVITY_NEW_TASK**标记： 当传递给startActivity()的Intent对象包含 FLAG_ACTIVITY_NEW_TASK标记时，系统会为需要启动的Activity寻找与当前Activity不同Task。如果要启动的Activity的Affinity属性与当前所有的Task的Affinity属性都不相同，系统会新建一个带那个Affinity属性的Task，并将要启动的Activity压到新建的Task栈中；否则将Activity压入那个Affinity属性相同的栈中。
>
> 2）**allowTaskReparenting**属性设置为true 如果一个activity的allowTaskReparenting属性为true， 那么它可以从一个Task（Task1）移到另外一个有相同Affinity的Task（Task2）中（Task2带到前台时）。 如果一个.apk文件从用户角度来看包含了多个"应用程序"，你可能需要对那些 Activity赋不同的Affinity值。

3）launchMode：

四个可选值，启动模式我们研究的核心，下面再详细讲! 他们分别是：**standard**(默认)，**singleTop**，**singleTask**，**singleInstance**

4）清空栈

> 当用户长时间离开Task（当前task被转移到后台）时，系统会清除task中栈底Activity外的所有Activity 。这样，当用户返回到Task时，只留下那个task最初始的Activity了。我们可以通过修改下面这些属性来 改变这种行为！
>
> **alwaysRetainTaskState**： 如果栈底Activity的这个属性被设置为true，上述的情况就不会发生。 Task中的所有activity将被长时间保存。
>
> **clearTaskOnLaunch** 如果栈底activity的这个属性被设置为true，一旦用户离开Task， 则 Task栈中的Activity将被清空到只剩下栈底activity。这种情况刚好与 alwaysRetainTaskState相反。即使用户只是短暂地离开，task也会返回到初始状态 （只剩下栈底acitivty）。
>
> **finishOnTaskLaunch** 与clearTaskOnLaunch相似，但它只对单独的activity操作，而不是整个Task。它可以结束任何Activity，包括栈底的Activity。 当它设置为true时，当前的Activity只在当前会话期间作为Task的一部分存在， 当用户退出Activity再返回时， 它将不存在。

4.Activity的四种加载模式详解：

接下来我们来详细地讲解下四种加载模式： 他们分别是：**standard**(默认)，**singleTop**，**singleTask**，**singleInstance** 在泡在网上的日子看到一篇图文并茂的讲解启动模式的，很赞，可能更容易理解吧，这里借鉴下：

原文链接：Activity启动模式图文详解：standard, singleTop, singleTask 以及 singleInstance

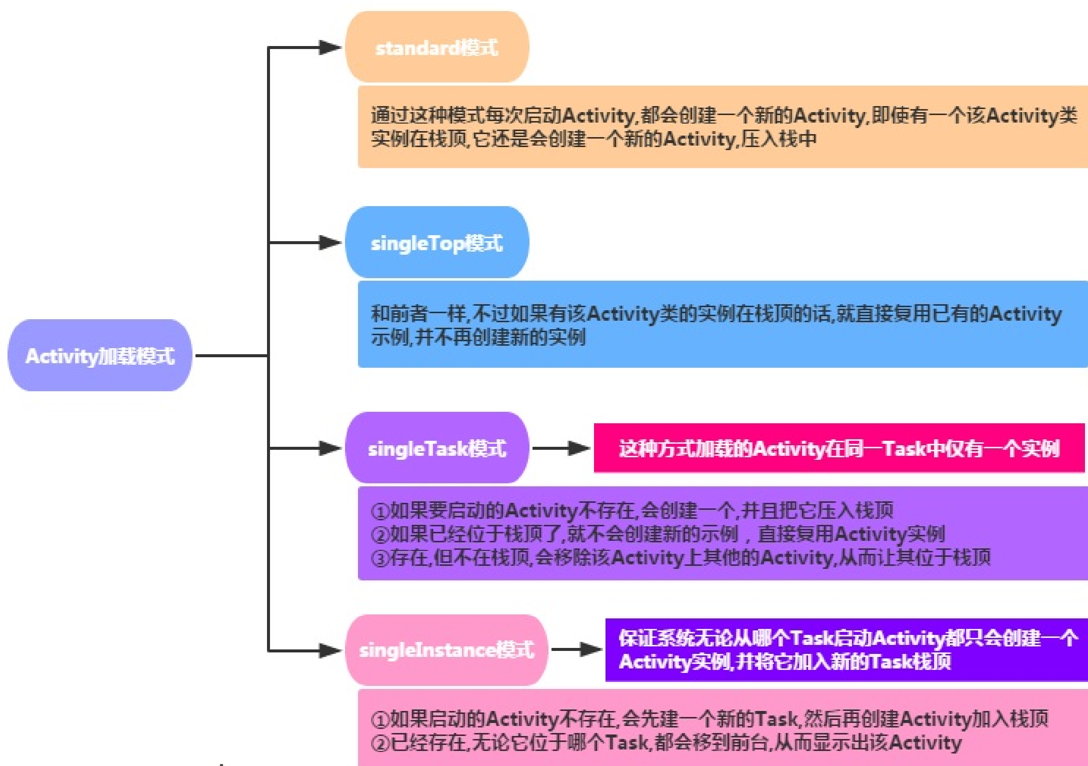
英文原文：Understand Android Activity's launchMode: standard, singleTop, singleTask and singleInstance 另外还有一篇详细讲解加载模式的：Android中Activity四种启动模式和taskAffinity属性详解

先来看看总结图：

Activity的四种加载模式

引言

Android采用Task来管理多个Activity,当我们启动App的时候,Android就会为我们创建一个Task,而第一个加入就是我们再AndroidManifest.xml中配置了MAIN和LAUNCHER的Activity,后续的依次压入栈中(PUSH),或者弹出(POP);但是我们却无法真正的去访问Task,我们只能够调用getTaskId( )方法来获取所在Task的ID,而Activity有四种加载模式我们可以在配置文件中设置属性:android:launchMode = "Xxx"设置为某个模式即可！



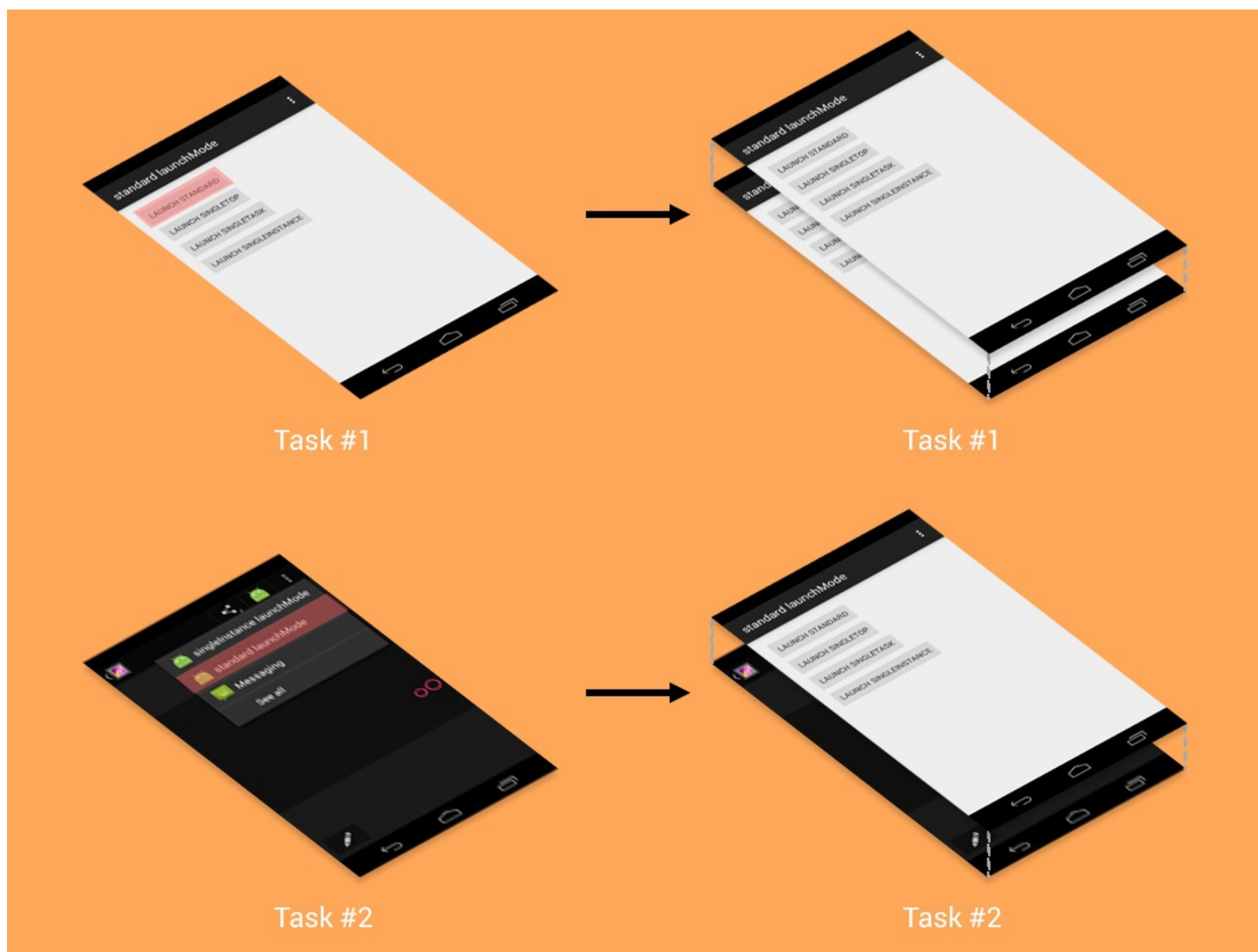


要注意:采用singleInstance的Activity总是位于栈顶;因为它所在的Task仅有它一个Activity

模式详解：

standard模式：

标准启动模式，也是activity的默认启动模式。在这种模式下启动的activity可以被多次实例化，即在同一个任务中可以存在多个activity的实例，每个实例都会处理一个Intent对象。如果Activity A的启动模式为standard，并且A已经启动，在A中再次启动Activity A，即调用startActivity（new Intent（this，A.class）），会在A的上面再次启动一个A的实例，即当前的栈中的状态为A-->A。



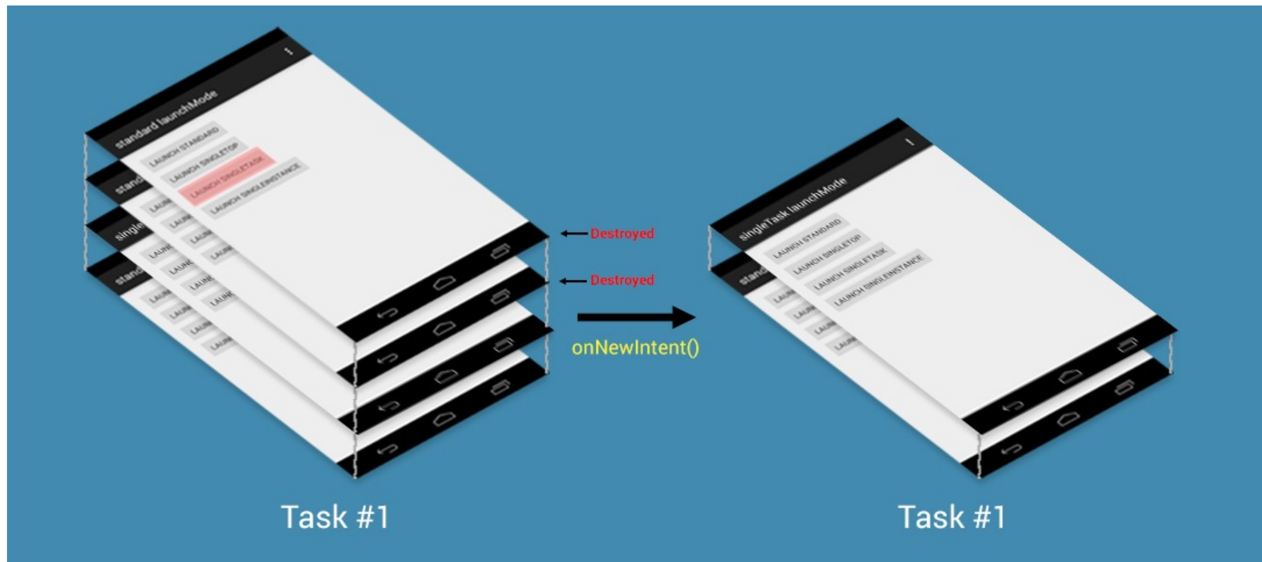
singleTop模式：

如果一个以singleTop模式启动的Activity的实例已经存在于任务栈的栈顶， 那么再启动这个Activity时，不会创建新的实例，而是重用位于栈顶的那个实例， 并且会调用该实例的**onNewIntent()**方法将Intent对象传递到这个实例中。 举例来说，如果A的启动模式为singleTop，并且A的一个实例已经存在于栈顶中， 那么再调用startActivity（new Intent（this，A.class））启动A时， 不会再次创建A的实例，而是重用原来的实例，并且调用原来实例的onNewIntent()方法。 这时任务栈中还是这有一个A的实例。如果以singleTop模式启动的activity的一个实例 已经存在与任务栈中，但是不在栈顶，那么它的行为和standard模式相同，也会创建多个实例。



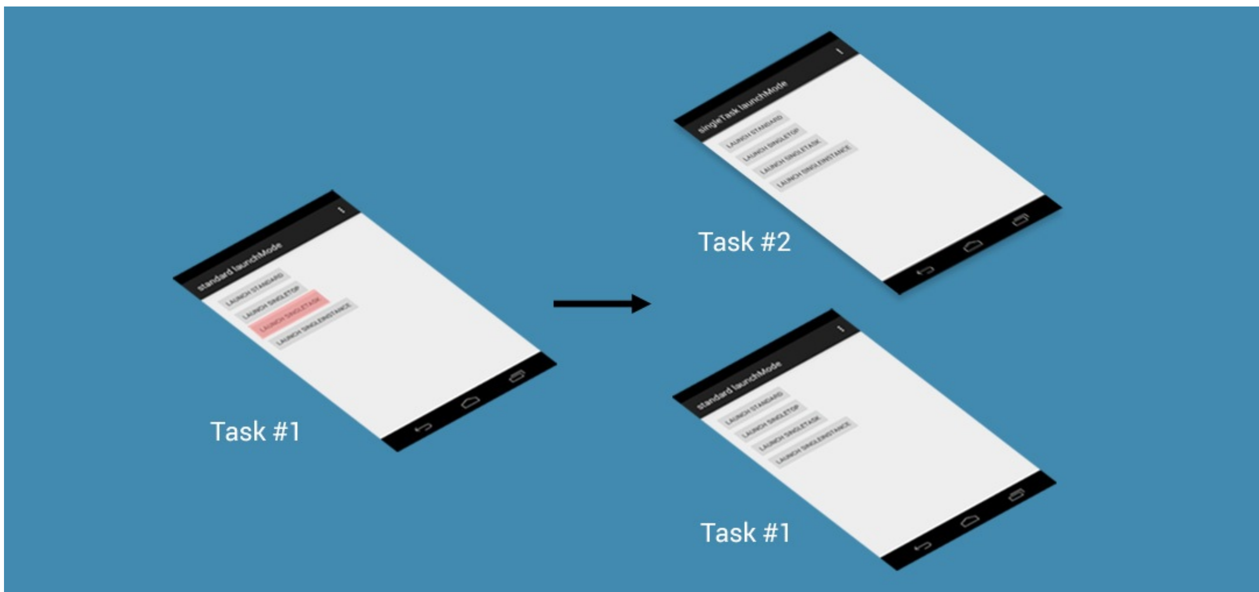
singleTask模式：

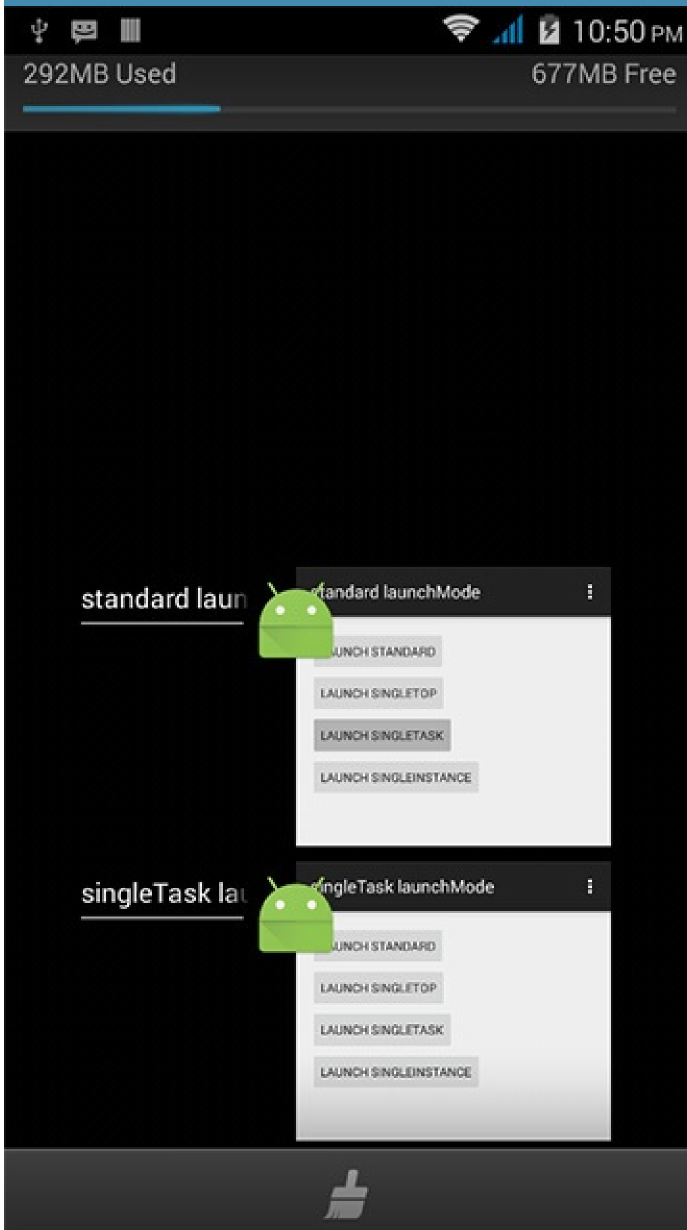
只允许在系统中有一个Activity实例。如果系统中已经有了一个实例， 持有这个实例的任务将移动到顶部，同时intent将被通过onNewIntent()发送。 如果没有，则会创建一个新的Activity并置放在合适的任务中。



官方文档中提到的一个问题：

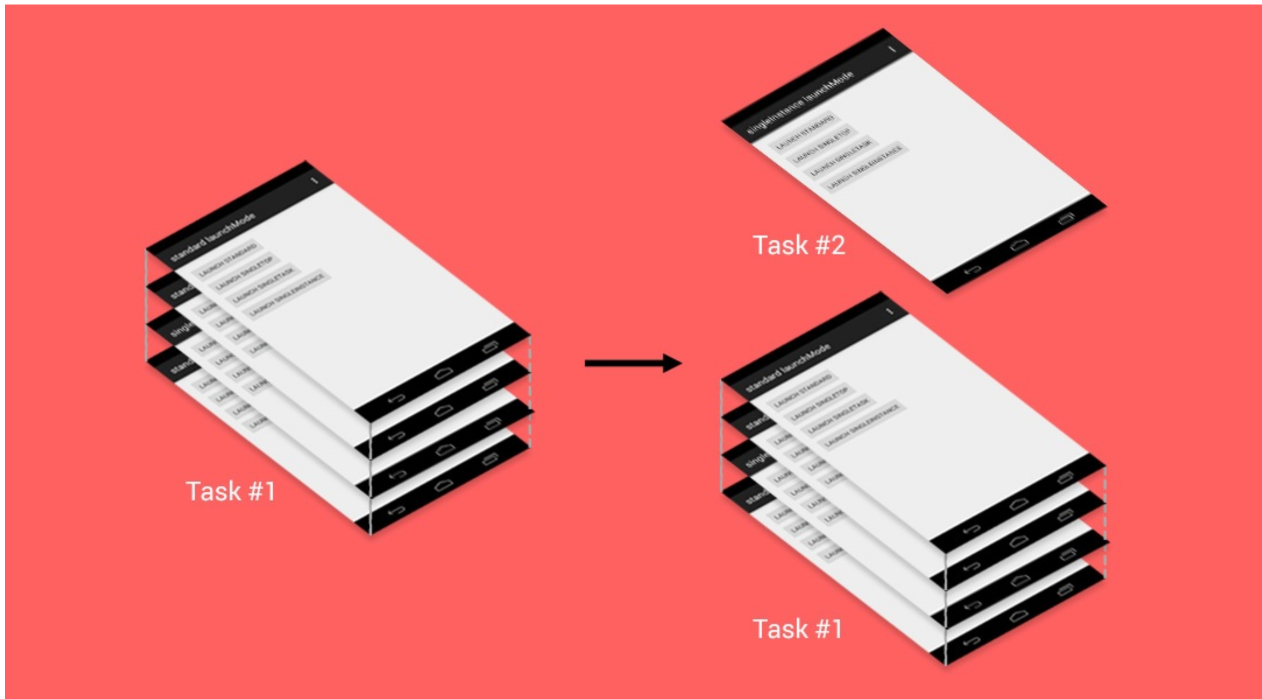
> 系统会创建一个新的任务，并将这个Activity实例化为新任务的根部（root） 这个则需要我们对taskAffinity进行设置了，使用taskAffinity后的解雇：





singleInstance模式

保证系统无论从哪个Task启动Activity都只会创建一个Activity实例,并将它加入新的Task栈顶 也就是说被该实例启动的其他activity会自动运行于另一个Task中。 当再次启动该activity的实例时，会重用已存在的任务和实例。并且会调用这个实例 的onNewIntent()方法，将Intent实例传递到该实例中。和singleTask相同， 同一时刻在系统中只会存在一个这样的Activity实例。



5.Activity拾遗

对于Activity可能有些东西还没讲到，这里预留一个位置，漏掉的都会在这里补上！首先是群友珠海-坤的建议，把开源中国的Activity管理类也贴上，嗯，这就贴上，大家可以直接用到 项目中~

1）开源中国客户端Activity管理类：

```
package net.oschina.app;

import java.util.Stack;

import android.app.Activity;
import android.app.ActivityManager;
import android.content.Context;

public class AppManager {

    private static Stack<Activity> activityStack;
    private static AppManager instance;

    private AppManager(){}
    /**
```

```
     * 单一实例
     */
    public static AppManager getAppManager(){
        if(instance==null){
            instance=new AppManager();
        }
        return instance;
    }
    /**
     * 添加Activity到堆栈
     */
    public void addActivity(Activity activity){
        if(activityStack==null){
            activityStack=new Stack<Activity>();
        }
        activityStack.add(activity);
    }
    /**
     * 获取当前Activity（堆栈中最后一个压入的）
     */
    public Activity currentActivity(){
        Activity activity=activityStack.lastElement();
        return activity;
    }
    /**
     * 结束当前Activity（堆栈中最后一个压入的）
     */
    public void finishActivity(){
        Activity activity=activityStack.lastElement();
        finishActivity(activity);
    }
    /**
     * 结束指定的Activity
     */
    public void finishActivity(Activity activity){
        if(activity!=null){
            activityStack.remove(activity);
            activity.finish();
            activity=null;
        }
    }
    /**
     * 结束指定类名的Activity
     */
    public void finishActivity(Class<?> cls){
        for (Activity activity : activityStack) {
            if(activity.getClass().equals(cls) ){
                finishActivity(activity);
            }
        }
    }
    /**
     * 结束所有Activity
```

```
     */
    public void finishAllActivity(){
        for (int i = 0, size = activityStack.size(); i < size; i++)
            if (null != activityStack.get(i)){
                activityStack.get(i).finish();
            }
        }
        activityStack.clear();
    }
    /**
     * 退出应用程序
     */
    public void AppExit(Context context) {
        try {
            finishAllActivity();
            ActivityManager activityMgr= (ActivityManager) context.
            activityMgr.restartPackage(context.getPackageName());
            System.exit(0);
        } catch (Exception e) {    }
    }
}
```

本节小结：

好的，本节就到这里，东西都比较苦涩难懂，暂时知道下即可，总结下Task进行整体调度的 相关操作吧：

> - 按Home键，将之前的Task切换到后台
> - 长按Home键，会显示出最近执行过的Task列表
> - 在Launcher或HomeScreen点击app图标，开启一个新Task，或者是将已有的Task调度到前台
> - 启动singleTask模式的Activity时，会在系统中搜寻是否已经存在一个合适的Task，若存在，则会将这个Task调度到前台以重用这个Task。如果这个Task中已经存在一个要启动的Activity的实例，则清除这个实例之上的所有Activity，将这个实例显示给用户。如果这个已存在的Task中不存在一个要启动的Activity的实例，则在这个Task的顶端启动一个实例。若这个Task不存在，则会启动一个新的Task，在这个新的Task中启动这个singleTask模式的Activity的一个实例。
> - 启动singleInstance的Activity时，会在系统中搜寻是否已经存在一个这个Activity的实例，如果存在，会将这个实例所在的Task调度到前台，重用这个Activity的实例（该Task中只有这一个Activity），如果不存在，会开启一个新任务，并在这个新Task中启动这个singleInstance模式的Activity的一个实例。

好的本节就到这里，关于Task与Activity加载模式的东西还是比较复杂的，下面给大家贴下编写该文的 时候的一些参考文献，可以自己看看~

参考文献：

- 1.Tasks and Back Stack
- 2.理解android中Activity和Task的关系
- 3.Activity启动模式图文详解：standard, singleTop, singleTask 以及 singleInstance
- 4.Understand Android Activity's launchMode: standard, singleTop, singleTask and singleInstance
- 5.Android中Activity四种启动模式和taskAffinity属性详解
- 6.Android的Activity和Tasks详解
- 7.Activity的四种启动模式和onNewIntent()
- 8.译：Android任务和返回栈完全解析，细数那些你所不知道的细节

4.2.1 Service初涉

本节引言

> 好的，我们在前三节中对Android中的Activity进行了研究学习，相信大家获益良多吧！本节开始我们继续来学习Android中的第二个组件：Service(服务)，好，废话不多说，开始本节内容！

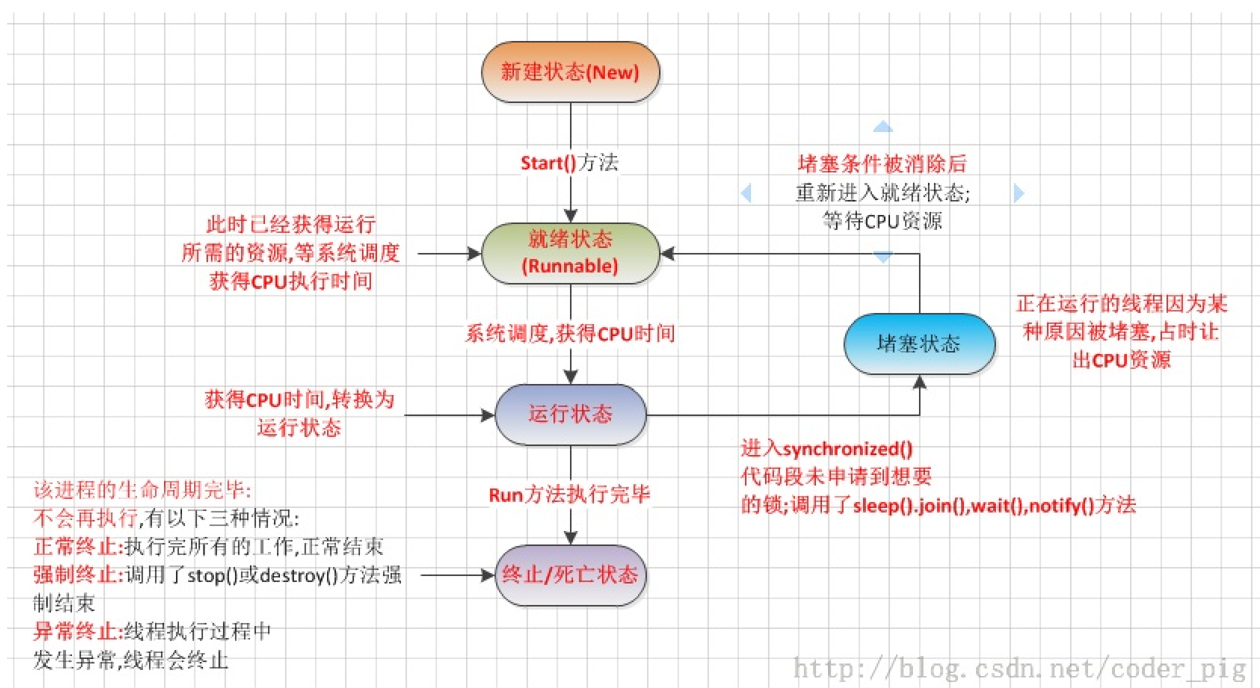
1.线程的相关概念

> 在开始学习Service之前我们先来了解下线程的一些概念！

1）相关概念：

> - 程序：为了完成特定任务，用某种语言编写的一组指令集合(一组静态代码)
> - 进程：运行中的程序，系统调度与资源分配的一个独立单位，操作系统会为每个进程分配一段内存空间！程序的依次动态执行，经历代码的加载，执行，执行完毕的完整过程！
> - 线程：比进程更小的执行单元，每个进程可能有多条线程，线程需要放在一个进程中才能执行，线程由程序负责管理，而进程则由系统进行调度！
> - 多线程的理解：并行执行多个条指令，将**CPU**时间片按照调度算法分配给各个线程，实际上是分时执行的，只是这个切换的时间很短，用户感觉到"同时"而已！

2）线程的生命周期：



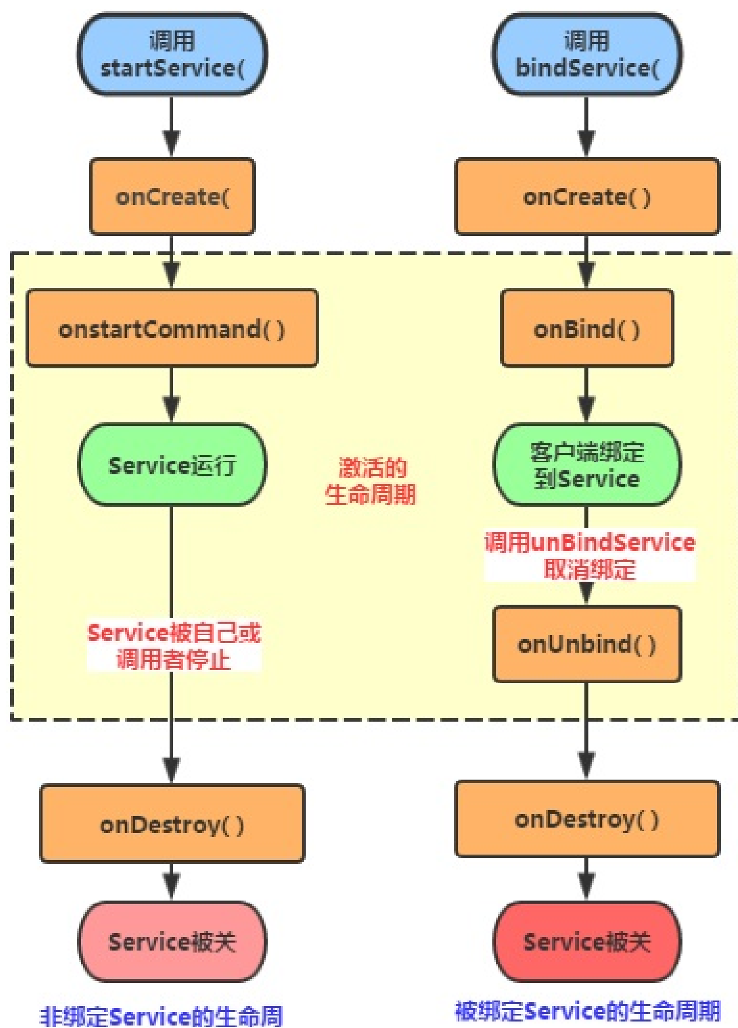
3）创建线程的三种方式：

> 1. 继承**Thread**类
> 2. 实现**Runnable**接口
> 3. 实现**Callable**接口 如果：使用的是2创建的线程的话，可以直接这样启动：
>
> ```
> new Thread(myThread).start();
> ```
>
> 当更多的时候我们喜欢使用匿名类，即下面这种写法：
>
> ```
> new Thread(new Runnable(){
>      public void run();
>          }).start();
> ```

2.Service与Thread线程的区别

> 其实他们两者并没有太大的关系，不过有很多朋友经常把这两个混淆了！Thread是线程，程序执行的最小单元，分配CPU的基本单位！ 而Service则是Android提供一个允许长时间留驻后台的一个组件，最常见的 用法就是做轮询操作！或者想在后台做一些事情，比如后台下载更新！ 记得别把这两个概念混淆！

3.Service的生命周期图



4.生命周期解析

好的，从上图的生命周期，我们可以知道，Android中使用Service的方式有两种：

> 1）**StartService()**启动**Service** 2）**BindService()**启动**Service** PS:还有一种，就是启动Service后，绑定Service！

1）相关方法详解：

> - **onCreate()**：当Service第一次被创建后立即回调该方法，该方法在整个生命周期 中只会调用依次！
> - **onDestory()**：当Service被关闭时会回调该方法，该方法只会回调一次！
> - **onStartCommand(intent,flag,startId)**：早期版本是onStart(intent,startId), 当客户端调用startService(Intent)方法时会回调，可多次调用StartService方法， 但不会再创建新的Service对象，而是继续复用前面产生的Service对象，但会继续回调 onStartCommand()方法！
> - **IBinder onOnbind(intent)**：该方法是Service都必须实现的方法，该方法会返回一个 IBinder对象，app通过该对象与Service组件进行通信！
> - **onUnbind(intent)**：当该Service上绑定的所有客户端都断开时会回调该方法！

2）StartService启动Service

> ①首次启动会创建一个Service实例,依次调用onCreate()和onStartCommand()方法,此时Service 进入运行状态,如果再次调用StartService启动Service,将不会再创建新的Service对象, 系统会直接复用前面创建的Service对象,调用它的onStartCommand()方法！ ②但这样的Service与它的调用者无必然的联系,就是说当调用者结束了自己的生命周期, 但是只要不调用stopService,那么Service还是会继续运行的! ③无论启动了多少次Service,只需调用一次StopService即可停掉Service

3）BindService启动Service

> ①当首次使用bindService绑定一个Service时,系统会实例化一个Service实例,并调用其onCreate()和onBind()方法,然后调用者就可以通过IBinder和Service进行交互了,此后如果再次使用bindService绑定Service,系统不会创建新的Sevice实例,也不会再调用onBind()方法,只会直接把IBinder对象传递给其他后来增加的客户端! ②如果我们解除与服务的绑定,只需调用unbindService(),此时onUnbind和onDestory方法将会被调用!这是一个客户端的情况,假如是多个客户端绑定同一个Service的话,情况如下 当一个客户完成和service之间的互动后，它调用 unbindService() 方法来解除绑定。当所有的客户端都和service解除绑定后，系统会销毁service。（除非service也被startService()方法开启） ③另外,和上面那张情况不同,bindService模式下的Service是与调用者相互关联的,可以理解为"一条绳子上的蚂蚱",要死一起死,在bindService后,一旦调用者销毁,那么Service也立即终止! 通过BindService调用Service时调用的Context的bindService的解析 **bindService**(Intent Service,ServiceConnection conn,int flags) **service**:通过该intent指定要启动的Service **conn**:ServiceConnection对象,用户监听访问者与Service间的连接情况, 连接成功回调该对象中的onServiceConnected(ComponentName,IBinder)方法; 如果Service所在的宿主由于异常终止或者其他原因终止,导致Service与访问者间断开 连接时调用**onServiceDisconnected**(CompanentName)方法,主动通过**unBindService()** 方法断开并不会调用上述方法! **flags**:指定绑定时是否自动创建**Service**(如果Service还未创建), 参数可以是0(不自动创建),BIND_AUTO_CREATE(自动创建)

4）StartService启动Service后bindService绑定

> 如果Service已经由某个客户端通过StartService()启动,接下来由其他客户端 再调用bindService()绑定到该Service后调用unbindService()解除绑定最后在 调用bindService()绑定到Service的话,此时所触发的生命周期方法如下:
> **onCreate( )->onStartCommand( )->onBind( )->onUnbind( )->onRebind( )**
> **PS:**前提是:onUnbind()方法返回true!!! 这里或许部分读者有疑惑了,调用了unbindService后Service不是应该调用 onDistory()方法么!其实这是因为这个Service是由我们的StartService来启动的 ,所以你调用onUnbind()方法取消绑定,Service也是不会终止的! 得出的结论**:** 假如我们使用bindService来绑定一个启动的Service,注意是已经启动的Service!!! 系统只是将Service的内部IBinder对象传递给Activity,并不会将Service的生命周期 与Activity绑定,因此调用unBindService( )方法取消绑定时,Service也不会被销毁！

5.生命周期验证

接下来我们写代码来验证下生命周期：

1）验证StartService启动Service的调用顺序

首先我们自定义一个Service,重写相关的方法,用户在logcat上打印验证:

TestService1.java

```
public class TestService1 extends Service {
    private final String TAG = "TestService1";
    //必须要实现的方法
    @Override
    public IBinder onBind(Intent intent) {
        Log.i(TAG, "onBind方法被调用!");
        return null;
    }

    //Service被创建时调用
    @Override
    public void onCreate() {
        Log.i(TAG, "onCreate方法被调用!");
        super.onCreate();
    }

    //Service被启动时调用
    @Override
    public int onStartCommand(Intent intent, int flags, int startI
        Log.i(TAG, "onStartCommand方法被调用!");
        return super.onStartCommand(intent, flags, startId);
    }

    //Service被关闭之前回调
    @Override
    public void onDestroy() {
        Log.i(TAG, "onDestory方法被调用!");
        super.onDestroy();
    }
}
```

AndroidManifest.xml完成**Service**注册

```
<!-- 配置Service组件,同时配置一个action -->
<service android:name=".TestService1">
            <intent-filter>
                <action android:name="com.jay.example.service.TEST_
            </intent-filter>
</service>
```

再接着是简单的布局文件,两个按钮,再最后是MainActivity的编写,在按钮的点击事件中分别 调用startService( )和stopService( )!

```
public class MainActivity extends Activity {

    private Button start;
    private Button stop;

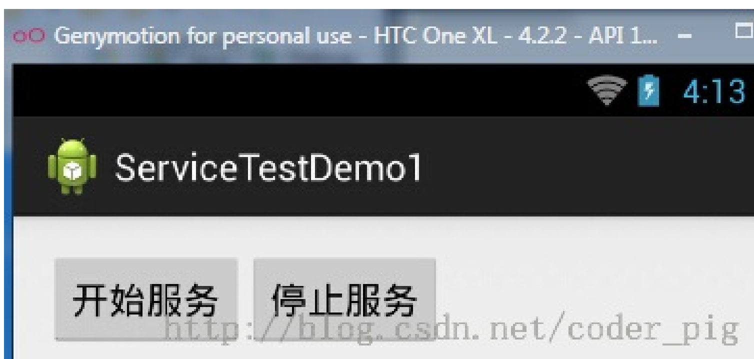
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        start = (Button) findViewById(R.id.btnstart);
        stop = (Button) findViewById(R.id.btnstop);
        //创建启动Service的Intent,以及Intent属性
        final Intent intent = new Intent();
        intent.setAction("com.jay.example.service.TEST_SERVICE1");
        //为两个按钮设置点击事件,分别是启动与停止service
        start.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                startService(intent);
            }
        });

        stop.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(intent);

            }
        });
    }
}
```

运行截图：



点击开始服务:

Application	Tag	Text
com.jay.example...	TestService1	onCreate方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!

吃饱饭没事做,点多几下:

com.jay.example...	TestService1	onCreate方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!

最后点击停止服务:

com.jay.example...	TestService1	onCreate方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!
com.jay.example...	TestService1	onStartCommand方法被调用!
com.jay.example...	TestService1	onDestory方法被调用!

结果分析:

从上面的运行结果我们可以验证我们生命周期图中解释的内容: 我们发现onBind()方法并没有被调用,另外多次点击启动Service,只会重复地调用onStartCommand 方法! 无论我们启动多少次Service,一个stopService就会停止Service!

2）验证BindService启动Service的顺序:

在开始讲写代码之前,我们先要来了解一些东西先: 首先是第一个大图下面给出的Context的bindService方法:

> - ServiceConnection对象:监听访问者与Service间的连接情况,如果成功连接,回调 onServiceConnected(),如果异常终止或者其他原因终止导致Service与访问者断开 连接则回调onServiceDisconnected方法,调用unBindService()不会调用该方法!
> - onServiceConnected方法中有一个IBinder对象,该对象即可实现与被绑定Service 之间的通信!我们再开发Service类时,默认需要实现IBinder onBind()方法,该方法返回的 IBinder对象会传到ServiceConnection对象中的onServiceConnected的参数,我们就可以 在这里通过这个IBinder与Service进行通信!

总结: **Step 1:**在自定义的Service中继承Binder,实现自己的IBinder对象 **Step 2:**通过onBind( )方法返回自己的IBinder对象 **Step 3:**在绑定该Service的类中定义一个ServiceConnection对象,重写两个方法, onServiceConnected和onDisconnected！然后直接读取IBinder传递过来的参数即可!

那么好了,接下来就是写代码验证了,这里的话我们定义一个用来计时的Service, 然后来演示BindService的用法以及方法调用流程!代码比较简单,不解释了!

TestService2.java:

```
public class TestService2 extends Service {
    private final String TAG = "TestService2";
    private int count;
    private boolean quit;

    //定义onBinder方法所返回的对象
    private MyBinder binder = new MyBinder();
    public class MyBinder extends Binder
    {
        public int getCount()
        {
            return count;
        }
    }

    //必须实现的方法,绑定改Service时回调该方法
    @Override
    public IBinder onBind(Intent intent) {
        Log.i(TAG, "onBind方法被调用!");
        return binder;
    }

    //Service被创建时回调
    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, "onCreate方法被调用!");
        //创建一个线程动态地修改count的值
        new Thread()
        {
            public void run()
            {
                while(!quit)
                {
                    try
                    {
                        Thread.sleep(1000);
                    }catch(InterruptedException e){e.printStackTrac
                    count++;
                }
            };
        }.start();

    }

    //Service断开连接时回调
    @Override
    public boolean onUnbind(Intent intent) {
        Log.i(TAG, "onUnbind方法被调用!");
        return true;
    }
```

```
    //Service被关闭前回调
    @Override
    public void onDestroy() {
        super.onDestroy();
        this.quit = true;
        Log.i(TAG, "onDestroyed方法被调用!");
    }

    @Override
    public void onRebind(Intent intent) {
        Log.i(TAG, "onRebind方法被调用!");
        super.onRebind(intent);
    }
}
```

在**AndroidManifest.xml**中对**Service**组件进行注册:

```
<service android:name=".TestService2" android:exported="false">
        <intent-filter>
            <action android:name="com.jay.example.service.TEST_SER\
        </intent-filter>
</service>
```

MainActivity.java:

```
public class MainActivity extends Activity {

    private Button btnbind;
    private Button btncancel;
    private Button btnstatus;

    //保持所启动的Service的IBinder对象,同时定义一个ServiceConnection对象
    TestService2.MyBinder binder;
    private ServiceConnection conn = new ServiceConnection() {

        //Activity与Service断开连接时回调该方法
        @Override
        public void onServiceDisconnected(ComponentName name) {
            System.out.println("------Service DisConnected-------")
        }

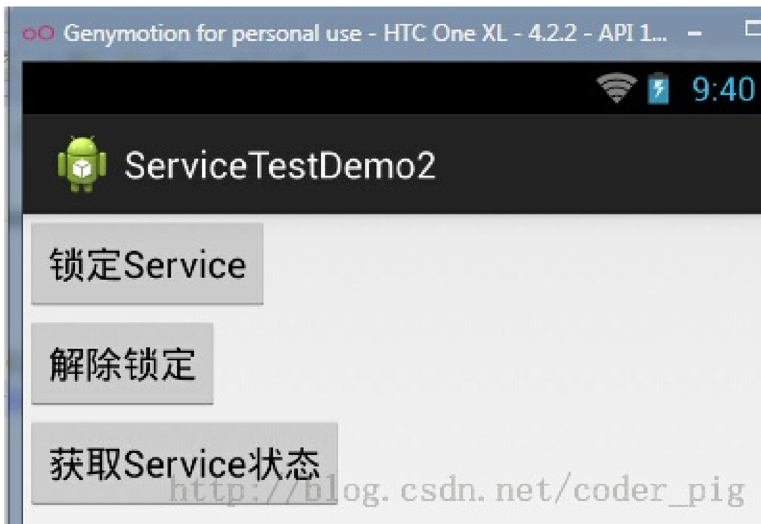
        //Activity与Service连接成功时回调该方法
        @Override
        public void onServiceConnected(ComponentName name, IBinder
            System.out.println("------Service Connected-------");
            binder = (TestService2.MyBinder) service;
        }
    };
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnbind = (Button) findViewById(R.id.btnbind);
        btncancel = (Button) findViewById(R.id.btncancel);
        btnstatus  = (Button) findViewById(R.id.btnstatus);
        final Intent intent = new Intent();
        intent.setAction("com.jay.example.service.TEST_SERVICE2");
        btnbind.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //绑定service
                bindService(intent, conn, Service.BIND_AUTO_CREATE)
            }
        });

        btncancel.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //解除service绑定
                unbindService(conn);
            }
        });

        btnstatus.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "Service的c
                        + binder.getCount(), Toast.LENGTH_SHORT).sh
            }
        });
    }
}
```

运行截图：



点击锁定**Service:**

```
EGL_emulation   eglSurfaceAttrib not implemented
TestService2    onCreate方法被调用!
TestService2    onBind方法被调用!
System.out      ------Service Connected-------
```

继续点击锁定:没任何变化

```
EGL_emulation   eglSurfaceAttrib not implemented
TestService2    onCreate方法被调用!
TestService2    onBind方法被调用!
System.out      ------Service Connected-------
```

获取当前**Service**的状态:

Service的count的值为:72

解除绑定:

```
TestService2    onCreate方法被调用!
TestService2    onBind方法被调用!
System.out      ------Service Connected---
TestService2    onUnbind方法被调用!
TestService2    onDestroyed方法被调用!
```

如果我们再绑定后直接关掉Activity的话会报错,然后会自动调用onUnbind和onDestory方法！

```
Activity com.jay.example.servicetestdemo2.MainActivity has leaked S
erviceConnection com.jay.example.servicetestdemo2.MainActivity$1@53
235ae4 that was originally bound here
android.app.ServiceConnectionLeaked: Activity com.jay.example.servi
cetestdemo2.MainActivity has leaked ServiceConnection com.jay.examp
le.servicetestdemo2.MainActivity$1@53235ae4 that was originally bou
nd here
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:560)
at dalvik.system.NativeStart.main(Native Method)
onUnbind方法被调用!
onDestroyed方法被调用!
```


。。。

从上面的运行结果验证了生命周期图中的:

> 使用BindService绑定Service,依次调用onCreate(),onBind()方法, 我们可以在onBind()方法中返回自定义的IBinder对象;再接着调用的是 ServiceConnection的onServiceConnected()方法该方法中可以获得 IBinder对象,从而进行相关操作;当Service解除绑定后会自动调用 onUnbind和onDestroyed方法,当然绑定多客户端情况需要解除所有 的绑定才会调用onDestoryed方法进行销毁哦！

4.2.2 Service进阶

本节引言

> 上节我们学习了Service的生命周期，以及两种启动Service的两种方法， 本节继续来深入了解Service中的IntentService，Service的使用实例： 前台服务与轮询的实现!

1.IntentService的使用

在上一节后我们已经知道了如何去定义和启动Service，但是如果我们直接把 耗时线程放到Service中的onStart()方法中， 虽然可以这样做，但是很容易 会引起ANR异常(Application Not Responding)， 而Android的官方在介绍 Service有下面这样一段话：

- A Service is **not** a separate process. The Service object itself does not imply it is running in its own process; unless otherwise specified, it runs in the same process as the application it is part of.
- A Service is **not** a thread. It is not a means itself to do work off of the main thread (to avoid Application Not Responding errors).

直接翻译：

> 1.Service不是一个单独的进程,它和它的应用程序在同一个进程中 2.Service不是一个线程,这样就意味着我们应该避免在Service中进行耗时操作

于是乎，Android给我们提供了解决上述问题的替代品，就是下面要讲的 **IntentService**； IntentService是继承与Service并处理异步请求的一个类,在IntentService中有 一个工作线程来处理耗时操作,请求的Intent记录会加入队列

工作流程：

> 客户端通过startService(Intent)来启动IntentService; 我们并不需要手动地区控制IntentService,当任务执行完后,IntentService会自动停止; 可以启动IntentService多次,每个耗时操作会以工作队列的方式在IntentService的onHandleIntent回调方法中执行,并且每次只会执行一个工作线程,执行完一，再到二这样!

再接着是代码演示,网上大部分的代码都是比较Service与IntentService的, 定义足够长的休眠时间,演示Service的ANR异常,然后引出IntentService有多好! 这里就不演示Service了,网上的都是自定义Service,然后在onStart()方法 中Thread.sleep(20000) 然后引发ANR异常,有兴趣的可以自己写代码试试, 这里的话只演示下IntentService的用法！

TestService3.java

```
public class TestService3 extends IntentService {
    private final String TAG = "hehe";
```

```
    //必须实现父类的构造方法
    public TestService3()
    {
        super("TestService3");
    }

    //必须重写的核心方法
    @Override
    protected void onHandleIntent(Intent intent) {
        //Intent是从Activity发过来的，携带识别参数，根据参数不同执行不同的任
        String action = intent.getExtras().getString("param");
        if(action.equals("s1"))Log.i(TAG,"启动service1");
        else if(action.equals("s2"))Log.i(TAG,"启动service2");
        else if(action.equals("s3"))Log.i(TAG,"启动service3");

        //让服务休眠2秒
        try{
            Thread.sleep(2000);
        }catch(InterruptedException e){e.printStackTrace();}
    }

    //重写其他方法,用于查看方法的调用顺序
    @Override
    public IBinder onBind(Intent intent) {
        Log.i(TAG,"onBind");
        return super.onBind(intent);
    }

    @Override
    public void onCreate() {
        Log.i(TAG,"onCreate");
        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId
        Log.i(TAG,"onStartCommand");
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void setIntentRedelivery(boolean enabled) {
        super.setIntentRedelivery(enabled);
        Log.i(TAG,"setIntentRedelivery");
    }

    @Override
    public void onDestroy() {
        Log.i(TAG,"onDestroy");
        super.onDestroy();
    }

}
```

AndroidManifest.xml注册下Service

```
<service android:name=".TestService3" android:exported="false">
    <intent-filter >
        <action android:name="com.test.intentservice"/>
    </intent-filter>
</service>
```

在**MainActivity**启动三次服务：

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Intent it1 = new Intent("com.test.intentservice");
        Bundle b1 = new Bundle();
        b1.putString("param", "s1");
        it1.putExtras(b1);

        Intent it2 = new Intent("com.test.intentservice");
        Bundle b2 = new Bundle();
        b2.putString("param", "s2");
        it2.putExtras(b2);

        Intent it3 = new Intent("com.test.intentservice");
        Bundle b3 = new Bundle();
        b3.putString("param", "s3");
        it3.putExtras(b3);

        //接着启动多次IntentService,每次启动,都会新建一个工作线程
        //但始终只有一个IntentService实例
        startService(it1);
        startService(it2);
        startService(it3);
    }
}
```

运行截图：

Application	Tag	Text
com.com.example...	hehe	onCreate
com.com.example...	hehe	onStartCommand
com.com.example...	hehe	onStartCommand
com.com.example...	hehe	onStartCommand
com.com.example...	hehe	启动service1
com.com.example...	hehe	启动service2
com.com.example...	hehe	启动service3
com.com.example...	hehe	onDestroy

小结：

> 当一个后台的任务,需要分成几个子任务,然后按先后顺序执行,子任务 (简单的说就是异步操作),此时如果我们还是定义一个普通Service然后 在onStart方法中开辟线程,然后又要去控制线程,这样显得非常的繁琐; 此时应该自定义一个IntentService然后再onHandleIntent()方法中完成相关任务！

2.Activity与Service通信

我们前面的操作都是通过Activity启动和停止Service，假如我们启动的是一个下载的后台Service，而我们想知道Service中下载任务的进度！那么这肯定是需要Service 与Activity进行通信的，而他们之间交流的媒介就是Service中的onBind()方法！ 返回一个我们自定义的Binder对象！

基本流程如下：

- 1.自定义Service中，自定义一个Binder类，然后将需要暴露的方法都写到该类中！
- 2.Service类中，实例化这个自定义Binder类，然后重写onBind()方法，将这个Binder对象返回！
- 3.Activity类中实例化一个ServiceConnection对象，重写onServiceConnected()方法，然后 获取Binder对象，然后调用相关方法即可！

3.一个简单前台服务的实现

学到现在，我们都知道Service一般都是运行在后来的，但是Service的系统优先级还是比较低的，当系统内存不足的时候，就有可能回收正在后台运行的Service，对于这种情况我们可以使用前台服务，从而让Service稍微没那么容易被系统杀死，当然还是有可能被杀死的...所谓的前台服务就是状态栏显示的Notification！

实现起来也很简单，最近做的项目刚好用到这个前台服务，就把核心的代码抠出来分享下：

在自定义的Service类中，重写onCreate()，然后根据自己的需求定制Notification；定制完毕后，调用startForeground(1,notification对象)即可！ 核心代码如下：

```
public void onCreate()
{
    super.onCreate();
    Notification.Builder localBuilder = new Notification.Builder(th
    localBuilder.setContentIntent(PendingIntent.getActivity(this, (
    localBuilder.setAutoCancel(false);
    localBuilder.setSmallIcon(R.mipmap.ic_cow_icon);
    localBuilder.setTicker("Foreground Service Start");
    localBuilder.setContentTitle("Socket服务端");
    localBuilder.setContentText("正在运行...");
    startForeground(1, localBuilder.getNotification());
}
```

运行效果截图：



4.简单定时后台线程的实现

除了上述的前台服务外，实际开发中Service还有一种常见的用法，就是执行定时任务， 比如轮询，就是每间隔一段时间就请求一次服务器，确认客户端状态或者进行信息更新 等！而Android中给我们提供的定时方式有两种使用Timer类与Alarm机

制！

> 前者不适合于需要长期在后台运行的定时任务，CPU一旦休眠，Timer中的定时任务 就无法运行；Alarm则不存在这种情况，他具有唤醒CPU的功能， 另外，也要区分CPU 唤醒与屏幕唤醒！

使用流程：

> - **Step 1**：获得**Service:** AlarmManager manager = (AlarmManager) getSystemService(ALARM_SERVICE);
> - **Step 2**：通过**set**方法设置定时任务 int anHour = 2 * 1000; long triggerAtTime = SystemClock.elapsedRealtime() + anHour; manager.set(AlarmManager.RTC_WAKEUP,triggerAtTime,pendingIntent);
> - **Step 3**：定义一个**Service** 在onStartCommand中开辟一条事务线程,用于处理一些定时逻辑
> - **Step 4**：定义一个**Broadcast(广播)**，用于启动**Service** 最后别忘了，在AndroidManifest.xml中对这Service与Boradcast进行注册！

参数详解： **set(int type,long startTime,PendingIntent pi)**

> **①type:** 有五个可选值: **AlarmManager.ELAPSED_REALTIME:** 闹钟在手机睡眠状态下不可用，该状态下闹钟使用相对时间（相对于系统启动开始），状态值为3; **AlarmManager.ELAPSED_REALTIME_WAKEUP** 闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟也使用相对时间，状态值为2；**AlarmManager.RTC** 闹钟在睡眠状态下不可用，该状态下闹钟使用绝对时间，即当前系统时间，状态值为1；**AlarmManager.RTC_WAKEUP** 表示闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟使用绝对时间，状态值为0; **AlarmManager.POWER_OFF_WAKEUP** 表示闹钟在手机关机状态下也能正常进行提示功能，所以是5个状态中用的最多的状态之一， 该状态下闹钟也是用绝对时间，状态值为4；不过本状态好像受SDK版本影响，某些版本并不支持；

PS:第一个参数决定第二个参数的类型,如果是REALTIME的话就用：SystemClock.elapsedRealtime( )方法可以获得系统开机到现在经历的毫秒数 如果是RTC的就用:System.currentTimeMillis()可获得从1970.1.1 0点到 现在做经历的毫秒数

> **②startTime**： 闹钟的第一次执行时间，以毫秒为单位，可以自定义时间，不过一般使用当前时间。 需要注意的是,本属性与第一个属性（type）密切相关,如果第一个参数对应的闹钟 使用的是相对时间（**ELAPSED_REALTIME**和**ELAPSED_REALTIME_WAKEUP**），那么本属 性就得使用相对时间（相对于系统启动时间来说）,比如当前时间就表示为: SystemClock.elapsedRealtime()；如果第一个参数对应的闹钟使用的是绝对时间 (RTC、RTC_WAKEUP、POWER_OFF_WAKEUP）,那么本属性就得使用绝对时间， 比如当前时间就表示为：System.currentTimeMillis()。
>
> **③PendingIntent:** 绑定了闹钟的执行动作，比如发送一个广播、给出提示等等。PendingIntent 是Intent的封装类。 需要注意的是，如果是通过启动服务来实现闹钟提示的话， PendingIntent对象的获取就应该采用Pending.getService (Context c,int i,Intent intent,int j)方法； 如果是通过广播来实现闹钟提示的话，PendingIntent对象的获取就应该采用 PendingIntent.getBroadcast (Context c,int i,Intent intent,int j)方法； 如果是采用Activity的方式来实现闹钟提示的话，PendingIntent对象的获取 就应该采用 PendingIntent.getActivity(Context c,int i,Intent intent,int j) 方法。 如果这三种方法错用了的话， 虽然不会报错，但是看不到闹钟提示效果。

另外**:**

从4.4版本后(API 19),Alarm任务的触发时间可能变得不准确,有可能会延时,是系统对于耗电性的优化,如果需要准确无误可以调用setExtra()方法~

核心代码：

```
public class LongRunningService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startIc
        //这里开辟一条线程,用来执行具体的逻辑操作:
        new Thread(new Runnable() {
            @Override
            public void run() {
                Log.d("BackService", new Date().toString());
            }
        }).start();
        AlarmManager manager = (AlarmManager) getSystemService(ALAR
        //这里是定时的,这里设置的是每隔两秒打印一次时间=-=,自己改
        int anHour = 2 * 1000;
        long triggerAtTime = SystemClock.elapsedRealtime() + anHour
        Intent i = new Intent(this,AlarmReceiver.class);
        PendingIntent pi = PendingIntent.getBroadcast(this, 0, i, (
        manager.set(AlarmManager.ELAPSED_REALTIME_WAKEUP, triggerAt
        return super.onStartCommand(intent, flags, startId);
    }
}
```

AlarmReceiver.java

```
public class AlarmReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Intent i = new Intent(context,LongRunningService.class);
        context.startService(i);
    }
}
```

本节小结：

本节我们继续对Service进行更深入的学习，IntentService以及Service 在实际开发中的两个常用的案例：前台Service的实现，以及Service后台 Service的实现！下一节中我们会继续研究Service的AIDL，跨进程通信， 敬请期待~

参考文献：《第一行代码 Android》—— 郭霖：很好的一本Android入门书！

4.2.3 Service精通

本节引言：

> 本节，我们继续来研究Service(服务)组件，本节将会学习下Android中的AIDL跨进程通信的一些 概念，并不深入到源码层次，暂时知道是什么，会用即可！开始本节内容~ 本节对应官方文档：Binder

1.Binder机制初涉

1）IBinder和Binder是什么鬼？

我们来看看官方文档怎么说：

Base interface for a remotable object, the core part of a lightweight remote procedure call mechanism designed for high performance when performing in-process and cross-process calls. This interface describes the abstract protocol for interacting with a remotable object. Do not implement this interface directly, instead extend from `Binder`.

The key IBinder API is `transact()` matched by `Binder.onTransact()`. These methods allow you to send a call to an IBinder object and receive a call coming in to a Binder object, respectively. This transaction API is synchronous, such that a call to `transact()` does not return until the target has returned from `Binder.onTransact()`; this is the expected behavior when calling an object that exists in the local process, and the underlying inter-process communication (IPC) mechanism ensures that these same semantics apply when going across processes.

The data sent through transact() is a `Parcel`, a generic buffer of data that also maintains some meta-data about its contents. The meta data is used to manage IBinder object references in the buffer, so that those references can be maintained as the buffer moves across processes. This mechanism ensures that when an IBinder is written into a Parcel and sent to another process, if that other process sends a reference to that same IBinder back to the original process, then the original process will receive the same IBinder object back. These semantics allow IBinder/Binder objects to be used as a unique identity (to serve as a token or for other purposes) that can be managed across processes.

The system maintains a pool of transaction threads in each process that it runs in. These threads are used to dispatch all IPCs coming in from other processes. For example, when an IPC is made from process A to process B, the calling thread in A blocks in transact() as it sends the transaction to process B. The next available pool thread in B receives the incoming transaction, calls Binder.onTransact() on the target object, and replies with the result Parcel. Upon receiving its result, the thread in process A returns to allow its execution to continue. In effect, other processes appear to use as additional threads that you did not create executing in your own process.

The Binder system also supports recursion across processes. For example if process A performs a transaction to process B, and process B while handling that transaction calls transact() on an IBinder that is implemented in A, then the thread in A that is currently waiting for the original transaction to finish will take care of calling Binder.onTransact() on the object being called by B. This ensures that the recursion semantics when calling remote binder object are the same as when calling local objects.

When working with remote objects, you often want to find out when they are no longer valid. There are three ways this can be determined:

- The `transact()` method will throw a `RemoteException` exception if you try to call it on an IBinder whose process no longer exists.
- The `pingBinder()` method can be called, and will return false if the remote process no longer exists.
- The `linkToDeath()` method can be used to register a `IBinder.DeathRecipient` with the IBinder, which will be called when its containing process goes away.

See Also

`Binder`

中文翻译：

> IBinder是远程对象的基本接口，是饿了高性能而设计的轻量级远程调用机制的核心部分。但他 不仅用于远程调用，也用于进程内调用。该接口定义了与远程对象间交互的协议。但不要直接实现 这个接口，而是继承(extends)**Binder**。
>
> IBinder主要的API是**transact()**，与之对应的API是**Binder.onTransact()**。通过前者，你能 想远程IBinder对象发送发出调用，后者使你的远程对象能够响应接收到的调用。IBinder的API都是 **Syncronous(同步)**执行的，比如**transact()**直到对方的**Binder.onTransact()**方法调用玩 后才返回。 调用发生在进程内时无疑是这样的，而在进程间时，在**IPC**的帮助下，也是同样的效果。
>
> 通过**transact()**发送的数据是**Parcel**，Parcel是一种一般的缓冲区，除了有数据外还带有 一些描述它内容的元数据。元数据用于管理IBinder对象的引用，这样就能在缓冲区从一个进程移动 到另一个进程时保存这些引用。这样就保证了当一个IBinder被写入到Parcel并发送到另一个进程中， 如果另一个进程把同一个IBinder的引用回发到原来的进程，那么这个原来的进程就能接收到发出的 那个IBinder的引用。这种机制使IBinder和Binder像唯一标志符那样在进程间管理。
>
> 系统为每个进程维护一个存放交互线程的线程池。这些交互线程用于派送所有从另外进程发来的IPC 调用。例如：当一个IPC从进程Ａ发到进程Ｂ，Ａ中那个发出调用的线程(这个应该不在线程池中)就阻塞 在**transact()**中了。进程Ｂ中的交互线程池中的一个线程接收了这个调用，它调用 **Binder.onTransact()**，完成后用一个Parcel来做为结果返回。然后进程Ａ中的那个等待的线程在 收到返回的Parcel后得以继续执行。实际上，另一个进程看起来就像是当前进程的一个线程， 但不是当前进程创建的。
>
> Binder机制还支持进程间的递归调用。例如，进程Ａ执行自己的IBinder的transact()调用进程Ｂ 的Binder，而进程Ｂ在其Binder.onTransact()中又用transact()向进程Ａ发起调用，那么进程Ａ 在等待它发出的调用返回的同时，还会用Binder.onTransact()响应进程Ｂ的transact()。 总之Binder造成的结果就是让我们感觉到跨进程的调用与进程内的调用没什么区别。
>
> 当操作远程对象时，你经常需要查看它们是否有效，有三种方法可以使用：
>
> - 1 transact()方法将在IBinder所在的进程不存在时抛出RemoteException异常。
> - 2 如果目标进程不存在，那么调用pingBinder()时返回false。
> - 3 可以用linkToDeath()方法向IBinder注册一个IBinder.DeathRecipient,在IBinder代表的进程退出时被调用。
>
> PS:中文翻译摘自：Android开发：什么是IBinder

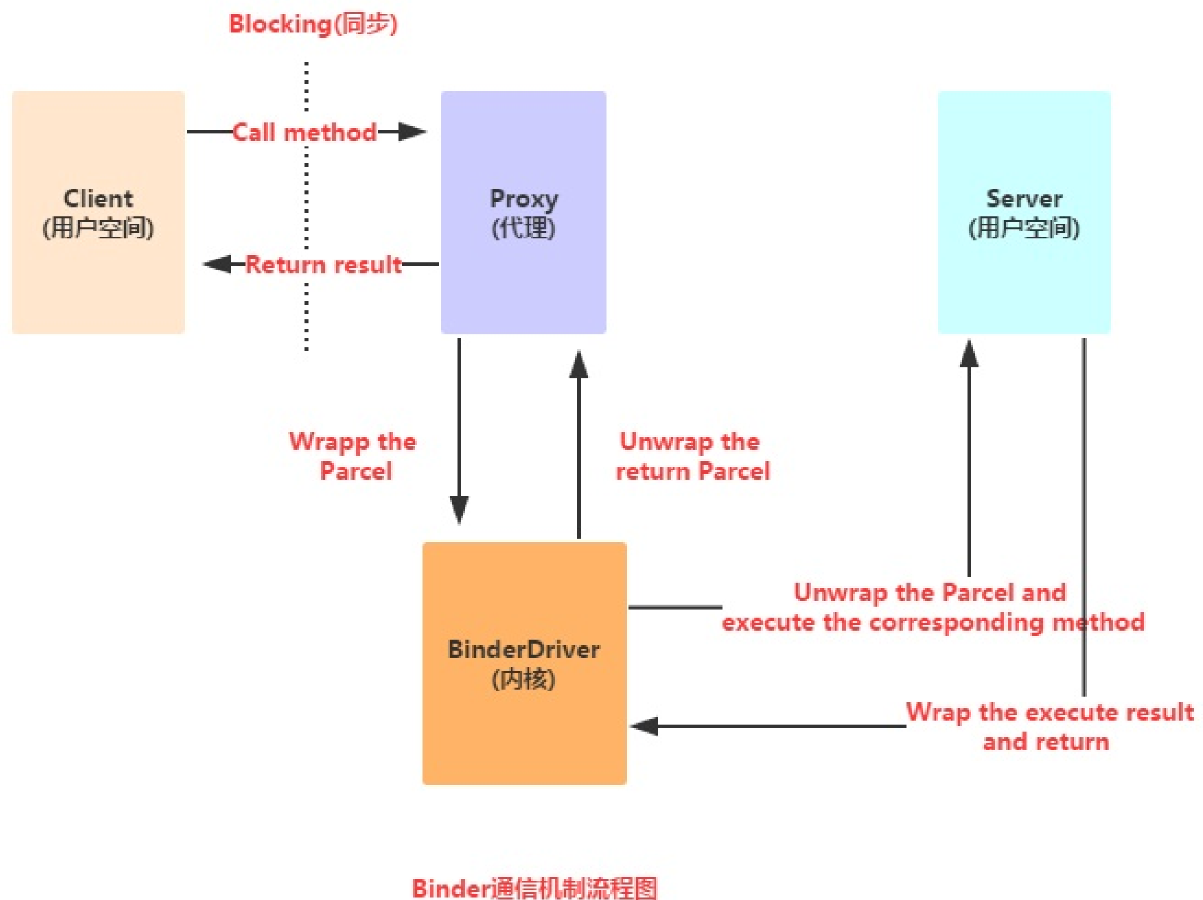
好吧，估计你看完上这一串东西可能云里雾里的，这里简单的小结下：

IBinder是Android给我们提供的一个进程间通信的一个接口，而我们一般是不直接实现这个接口的， 而是通过继承**Binder**类来实现进程间通信！是**Android**中实现**IPC(**进程间通信**)**的一种方式！

2）Binder机制浅析

> Android中的Binder机制由一系列系统组件构成： **Client**、**Server**、**Service Manager**和**Binder**驱动程序

大概调用流程如下，另外Service Manager比较复杂，这里并不详细研究！



Binder通信机制流程图

流程解析：

> -> Client调用某个代理接口中的方法时，代理接口的方法会将Client传递的参数打包成Parcel对象； -> 然后代理接口把该Parcel对象发送给内核中的Binder driver；； -> 然后Server会读取Binder Driver中的请求数据，假如是发送给自己的，解包Parcel对象， 处理并将结果返回； PS:代理接口中的定义的方法和Server中定义的方法是一一对应的， 另外，整个调用过程是一个同步的，即Server在处理时，Client会被Block(锁)住! 而这里说的代理接口的定义就是等下要说的**AIDL**(Android接口描述语言)！

3）为何Android使用Binder机制来实现进程间的通信？

> 1. 可靠性：在移动设备上，通常采用基于Client-Server的通信方式来实现互联网与设备间的内部通信。目前linux支持IPC包括传统的管道，System V IPC，即消息队列/共享内存/信号量，以及socket中只有socket支持Client-Server的通信方式。Android系统为开发者提供了丰富进程间通信的功能接口，媒体播放，传感器，无线传输。这些功能都由不同的server来管理。开发都只关心将自己应用程序的client与server的通信建立起来便可以使用这个服务。毫无疑问，如若在底层架设一套协议来实现Client-Server通信，增加了系统的复杂性。在资源有限的手机 上来实现这种复杂的环境，可靠性难以保证。
> 2. 传输性能：socket主要用于跨网络的进程间通信和本机上进程间的通信，但传输效率低，开销大。消息队列和管道采用存储-转发方式，即数据先从发送方缓存区拷贝到内核开辟的一块缓存区中，然后从内核缓存区拷贝到接收方缓存区，其过程至少有两次拷贝。虽然共享内存无需拷贝，但控制复杂。比较各种IPC方式的数据拷贝次数。共享内存：0次。Binder：1次。Socket/管道/消息队列：2次。
> 3. 安全性：Android是一个开放式的平台，所以确保应用程序安全是很重要的。Android对每一个安装应用都分配了UID/PID,其中进程的UID是可用来鉴别进程身份。传统的只能由用户在数据包里填写UID/PID，这样不可靠，容易被恶意程序利用。而我们要求由内核来添加可靠的UID。 所以，出于可靠性、传输性、安全性。android建立了一套新的进程间通信方式。——摘自:Android中的Binder机制的简要理解

当然，作为一个初级的开发者我们并不关心上述这些，Binder机制给我们带来的最直接的好处就是： 我们无需关心底层如何实现，只需按照**AIDL**的规则，自定义一个接口文件， 然后调用调用接口中的方法，就可以完成两个进程间的通信了！

2.AIDL使用详解

1）AIDL是什么？

> 嘿嘿，前面我们讲到**IPC**这个名词，他的全名叫做：跨进程通信**(interprocess communication)**， 因为在Android系统中,个个应用程序都运行在自己的进程中,进程之间一般是无法直接进行数据交换的, 而为了实现跨进程，Android给我们提供了上面说的Binder机制，而这个机制使用的接口语言就是: **AIDL**(Android Interface Definition Language)，他的语法很简单，而这种接口语言并非真正的编程 语言，只是定义两个进程间的通信接口而已！而生成符合通信协议的Java代码则是由Android SDK的 platform-tools目录下的**aidl.exe**工具生成，生成对应的接口文件在:gen目录下，一般是:Xxx.java的接口！ 而在该接口中包含一个**Stub**的内部类， 该类中实现了在该类中实现了**IBinder**接口与自定义的通信接口, 这个类将会作为远程Service的回调类——实现了IBinder接口,所以可作为Service的onBind( )方法的返回值！

2）AIDL实现两个进程间的简单通信

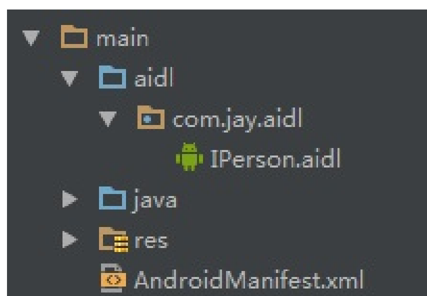
在开始编写AIDL接口文件前，我们需要了解下编写AIDL的一些注意事项：

AIDL注意事项：

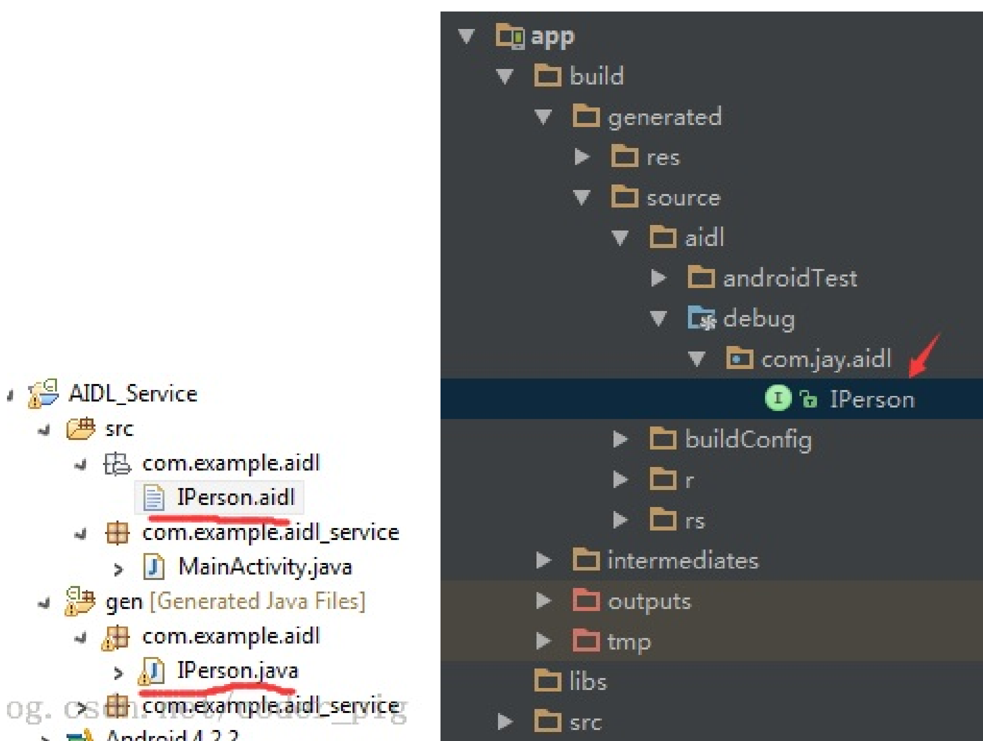
- 接口名词需要与aidl文件名相同
- 接口和方法前面不要加访问权限修饰符：public ,private,protected等，也不能用static final!
- AIDL默认支持的类型包括Java基本类型，**String**，**List**，**Map**，**CharSequence**，除此之外的其他类型都 需要import声明，对于使用自定义类型作为参数或者返回值，自定义类型需要实现Parcelable接口， 详情请看后面的传递复杂数据类型
- 自定义类型和AIDL生成的其它接口类型在aidl描述文件中，应该显式import，即便在该类和定义 的包在同一个包中。

另外，如果编写aidl你用的编译器是:Eclipse的话要注意： 不要直接new file然后建立哦!这样的话是打不开文件,从而不能编写代码哦！ ①直接新建一个txt文件,编写好后保存为.aidl格式,然后复制到对应路径下 ②因为aidl和接口类似,所以直接new interface,编写好内容后,来到对应java文件所在目录下修改文件后缀名;

假如你使用的是Android Studio的话，不同于Eclipse，如果你按照Eclipse那样创建一个AIDL文件，会发现 并没有编译生成对应的XXX.java文件，AS下创建AIDL需要在main目录下新建一个aidl文件夹，然后定义一个 和aidl包名相同的包，最后创建一个aidl文件，接着按ctrl + f9重新编译，就可以了！



上面两者成功编译的结果如下，你可以分别在对应目录下找到对应的AIDL文件



1.服务端：

Step 1：创建AIDL文件：

IPerson.aidl

```
package com.jay.aidl;

interface IPerson {
    String queryPerson(int num);
}
```

我们打开**IPerson.java**看看里面的代码：

IPerson.java

```
/*
 * This file is auto-generated.  DO NOT MODIFY.
 * Original file: C:\\Code\\ASCode\\AIDLServer\\app\\src\\main\\aic
 */
package com.jay.aidl;
public interface IPerson extends android.os.IInterface
{
/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder impleme
{
private static final java.lang.String DESCRIPTOR = "com.jay.aidl.IF
/** Construct the stub at attach it to the interface. */
public Stub()
{
this.attachInterface(this, DESCRIPTOR);
}
/**
 * Cast an IBinder object into an com.jay.aidl.IPerson interface,
 * generating a proxy if needed.
 */
public static com.jay.aidl.IPerson asInterface(android.os.IBinder c
{
if ((obj==null)) {
return null;
}
android.os.IInterface iin = obj.queryLocalInterface(DESCRIPTOR);
if (((iin!=null)&&(iin instanceof com.jay.aidl.IPerson))) {
return ((com.jay.aidl.IPerson)iin);
}
return new com.jay.aidl.IPerson.Stub.Proxy(obj);
}
@Override public android.os.IBinder asBinder()
{
return this;
}
```

```
@Override public boolean onTransact(int code, android.os.Parcel dat
{
switch (code)
{
case INTERFACE_TRANSACTION:
{
reply.writeString(DESCRIPTOR);
return true;
}
case TRANSACTION_queryPerson:
{
data.enforceInterface(DESCRIPTOR);
int _arg0;
_arg0 = data.readInt();
java.lang.String _result = this.queryPerson(_arg0);
reply.writeNoException();
reply.writeString(_result);
return true;
}
}
return super.onTransact(code, data, reply, flags);
}
private static class Proxy implements com.jay.aidl.IPerson
{
private android.os.IBinder mRemote;
Proxy(android.os.IBinder remote)
{
mRemote = remote;
}
@Override public android.os.IBinder asBinder()
{
return mRemote;
}
public java.lang.String getInterfaceDescriptor()
{
return DESCRIPTOR;
}
@Override public java.lang.String queryPerson(int num) throws andro
{
android.os.Parcel _data = android.os.Parcel.obtain();
android.os.Parcel _reply = android.os.Parcel.obtain();
java.lang.String _result;
try {
_data.writeInterfaceToken(DESCRIPTOR);
_data.writeInt(num);
mRemote.transact(Stub.TRANSACTION_queryPerson, _data, _reply, 0);
_reply.readException();
_result = _reply.readString();
}
finally {
_reply.recycle();
_data.recycle();
}
```

```
return _result;
}
}
static final int TRANSACTION_queryPerson = (android.os.IBinder.FIRS
}
public java.lang.String queryPerson(int num) throws android.os.Remo
}
```

这里我们关注的只是**asInterface(IBinder)**和我们定义的接口中的**queryPerson()**方法!

该方法会把IBinder类型的对象转换成IPerson类型的,必要时生成一个代理对象返回结果！

其他的我们可以不看，直接跳过，进行下一步。

Step 2：自定义我们的Service类,完成下述操作:**

1)继承Service类,同时也自定义了一个PersonQueryBinder类用来继承**IPerson.Stub**类 就是实现了**IPerson**接口和**IBinder**接口

2)实例化自定义的Stub类,并重写Service的onBind方法,返回一个binder对象!

AIDLService.java

```
package com.jay.aidlserver;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;
import com.jay.aidl.IPerson.Stub;

/**
 * Created by Jay on 2015/8/18 0018.
 */
public class AIDLService extends Service {

    private IBinder binder = new PersonQueryBinder();
    private String[] names = {"B神","艹神","基神","J神","翔神"};

    private String query(int num)
    {
        if(num > 0 && num < 6){
            return names[num - 1];
        }
        return null;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    private final class PersonQueryBinder extends Stub{
        @Override
        public String queryPerson(int num) throws RemoteException {
            return query(num);
        }
    }
}
```

Step 3：在AndroidManifest.xml文件中注册Service

```
<service android:name=".AIDLService">
            <intent-filter>
                <action android:name="android.intent.action.AIDLSer
                <category android:name="android.intent.category.DEF
            </intent-filter>
        </service>
```

这里我们并没有提供Activity界面，但是改应用提供的Service可以供其他app来调用！

2.客户端 直接把服务端的那个aidl文件复制过来，然后我们直接在MainActivity中完成，和绑定本地Service的操作 有点类似，流程如下： 1)自定义PersonConnection类实现**ServiceConnection**接口 2)以PersonConnection对象作为参数,调用bindService绑定远程Service **bindService(service,conn,BIND_AUTO_CREATE);** ps:第三个参数是设置如果服务没有启动的话,自动创建 3)和本地Service不同，绑定远程**Service**的**ServiceConnection**并不能直接获取**Service**的**onBind( )**方法 返回的IBinder对象，只能返回**onBind( )**方法所返回的代理对象，需要做如下处理: **iPerson = IPerson.Stub.asInterface(service);** 再接着完成初始化,以及按钮事件等就可以了

具体代码如下：

MainActivity.java

```
package com.jay.aidlclient;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.jay.aidl.IPerson;

public class MainActivity extends AppCompatActivity implements View

    private EditText edit_num;
    private Button btn_query;
    private TextView txt_name;
    private IPerson iPerson;
    private PersonConnection conn = new PersonConnection();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
        //绑定远程Service
        Intent service = new Intent("android.intent.action.AIDLServ
        service.setPackage("com.jay.aidlserver");

        bindService(service, conn, BIND_AUTO_CREATE);
        btn_query.setOnClickListener(this);
    }

    private void bindViews() {
```

```
        edit_num = (EditText) findViewById(R.id.edit_num);
        btn_query = (Button) findViewById(R.id.btn_query);
        txt_name = (TextView) findViewById(R.id.txt_name);
    }

    @Override
    public void onClick(View v) {
        String number = edit_num.getText().toString();
        int num = Integer.valueOf(number);
        try {
            txt_name.setText(iPerson.queryPerson(num));
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        edit_num.setText("");
    }

    private final class PersonConnection implements ServiceConnecti
        public void onServiceConnected(ComponentName name, IBinder 
            iPerson = IPerson.Stub.asInterface(service);
        }
        public void onServiceDisconnected(ComponentName name) {
            iPerson = null;
        }
    }
}
```

接下来先启动AIDLServivce，然后再启动AIDLClient，输入查询序号，即可获得对应姓名！当然也可以直接启动AIDLClient，也会获得同样效果：

效果图如下：



3）传递复杂数据的AIDL Service

> 上面的例子我们传递的只是要给int类型的参数，然后服务端返回一个String类型的参数，看似满足 我们的基本需求，不过实际开发中，我们可能需要考虑传递复杂数据类型的情况！下面我们来学习下 如何向服务端传递复杂数据类型的数据！开始之前我们先来了解**Parcelable**接口！

——Parcelable接口简介：

相信用过序列化的基本上都知道这个接口了，除了他还有另外一个Serializable，同样是用于序列化的， 只是Parcelable更加轻量级，速度更快！但是写起来就有点麻烦了，当然如果你用的as的话可以用 的插件来完成序列化，比如：**Android Parcelable Code Generator** 当然，这里我们还是手把手教大家来实现这个接口~

首先需要实现：**writeToParcel**和**readFromPacel**方法 写入方法将对象写入到包裹(parcel)中,而读取方法则从包裹中读取对象, 请注意,写入属性顺序需与读取顺序相同

接着需要在：该类中添加一个名为**CREATOR**的**static final**属性 改属性需要实现：android.os.Parcelable.Creator<t>接口</t>

再接着需要从写接口中的两个方法： **createFromParcel**(Parcel source)方法:实现从source创建出JavaBean实例的功能 **newArray**(int size):创建一个类型为T,长度为size的数组,只有一个简单的return new T[size]; (这里的T是Person类)

最后，**describeContents()**:这个我也不知道是拿来干嘛的,直接返回0即可！不用理他

——另外，非原始类型中，除了**String**和**CharSequence**以外，其余均需要一个方向指示符。 方向指示符包括 **in**、**out**、 和**inout**。in表示由客户端设置，out表示由服务端设置，inout表示客户端和服务端都设置了该值。

好的，接着来写代码试试(AS这里自定义类型有点问题，暂时还没解决，就用回Eclipse~)：

代码示例：

自定义两种对象类型:Person与Salary,Person作为调用远程的Service的参数,Salary作为返回值! 那么首先要做的就是创建Person与Salary类,同时需要实现Parcelable接口

1.——服务端

Step 1：创建Person.aidl和Salary.aidl的文件，因为他们需要实现Parcelable接口，所以就下面一条语句：

```
Person.aidl:     parcelable Person;
Salary.aidl:     parcelable Salary;
```

Step 2：分别建立Person类与Salary类，需实现Parcelable接口，重写对应的方法!

> PS:因为我们后面是根据Person对象来获取Map集合中的数据,所以Person.java中我们重写了hashcode和equals 的方法;而Salary类则不需要!

Person.java:

```
package com.jay.example.aidl;

import android.os.Parcel;
import android.os.Parcelable;

/**
 * Created by Jay on 2015/8/18 0018.
 */
public class Person implements Parcelable{

    private Integer id;
    private String name;

    public Person() {}

    public Person(Integer id, String name) {
        this.id = id;
        this.name = name;
    }

    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    //实现Parcelable必须实现的方法,不知道拿来干嘛的,直接返回0就行了
    @Override
    public int describeContents() {
        return 0;
    }

    //写入数据到Parcel中的方法
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        //把对象所包含的数据写入到parcel中
        dest.writeInt(id);
        dest.writeString(name);
    }

    //必须提供一个名为CREATOR的static final属性 该属性需要实现
    //android.os.Parcelable.Creator<T>接口
    public static final Parcelable.Creator<Person> CREATOR = new Pa
        //从Parcel中读取数据,返回Person对象
        @Override
        public Person createFromParcel(Parcel source) {
            return new Person(source.readInt(),source.readString()
        }
        @Override
        public Person[] newArray(int size) {
            return new Person[size];
        }
    };

    //因为我们集合取出元素的时候是根据Person对象来取得,所以比较麻烦,
    //需要我们重写hashCode()和equals()方法
    @Override
    public int hashCode()
    {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((name == null) ? 0 : name.hashCo
        return result;
    }
    @Override
    public boolean equals(Object obj)
```

```
    {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Person other = (Person) obj;
        if (name == null)
        {
            if (other.name != null)
                return false;
        }
        else if (!name.equals(other.name))
            return false;
        return true;
    }
}
<pre><p><strong>Salary.java</strong>~照葫芦画瓢</p>

<pre>
package com.jay.example.aidl;

import android.os.Parcel;
import android.os.Parcelable;

/**
 * Created by Jay on 2015/8/18 0018.
 */
public class Salary implements Parcelable {

    private String type;
    private Integer salary;

    public Salary() {
    }

    public Salary(String type, Integer salary) {
        this.type = type;
        this.salary = salary;
    }

    public String getType() {
        return type;
    }

    public Integer getSalary() {
        return salary;
    }

    public void setType(String type) {
        this.type = type;
    }
```

```
    public void setSalary(Integer salary) {
        this.salary = salary;
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(type);
        dest.writeInt(salary);
    }

    public static final Parcelable.Creator<Salary> CREATOR = new Pa
        //从Parcel中读取数据,返回Person对象
        @Override
        public Salary createFromParcel(Parcel source) {
            return new Salary(source.readString(), source.readInt(
        }

        @Override
        public Salary[] newArray(int size) {
            return new Salary[size];
        }
    };

    public String toString() {
        return "工作:" + type + "    薪水: " + salary;
    }
}
```

Step 3：创建一个ISalary.aidl的文件，在里面写一个简单的获取工资信息的方法：

```
package com.jay.example.aidl;

import com.jay.example.aidl.Salary;
import com.jay.example.aidl.Person;
interface ISalary
{
    //定义一个Person对象作为传入参数
    //接口中定义方法时,需要制定新参的传递模式,这里是传入,所以前面有一个in
    Salary getMsg(in Person owner);
}
```

ps:这里可以记得如果使用的是自定义的数据类型的话,需要import哦！！！切记！！！

Step 4：核心Service的编写： 定义一个SalaryBinder类继承Stub,从而实现ISalary和IBinder接口;定义一个存储信息的Map集合! 重新onBind方法,返回SalaryBinder类的对象实例!

AidlService.java

```
package com.jay.example.aidl_complexservice;

import java.util.HashMap;
import java.util.Map;
import com.jay.example.aidl.ISalary.Stub;
import com.jay.example.aidl.Person;
import com.jay.example.aidl.Salary;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;

public class AidlService extends Service {

    private SalaryBinder salaryBinder;
    private static Map<Person,Salary> ss = new HashMap<Person, Sala
    //初始化Map集合,这里在静态代码块中进行初始化,当然你可也以在构造方法中完成
    static
    {
        ss.put(new Person(1, "Jay"), new Salary("码农", 2000));
        ss.put(new Person(2, "GEM"), new Salary("歌手", 20000));
        ss.put(new Person(3, "XM"), new Salary("学生", 20));
        ss.put(new Person(4, "MrWang"), new Salary("老师", 2000));
    }

    @Override
    public void onCreate() {
        super.onCreate();
        salaryBinder = new SalaryBinder();
    }

    @Override
    public IBinder onBind(Intent intent) {
        return salaryBinder;
    }

    //同样是继承Stub,即同时实现ISalary接口和IBinder接口
    public class SalaryBinder extends Stub
    {
        @Override
        public Salary getMsg(Person owner) throws RemoteException
            return ss.get(owner);
        }
```

```
    }

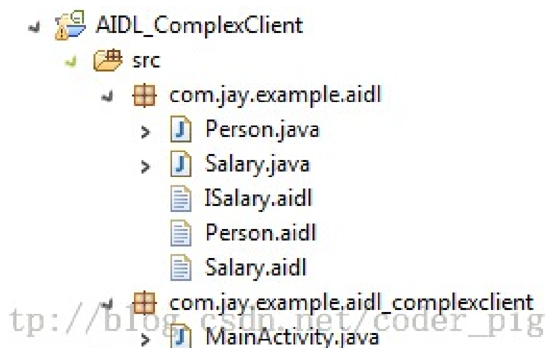
    @Override
    public void onDestroy() {
        System.out.println("服务结束！");
        super.onDestroy();
    }
}
```

注册下**Service:**

```
<service android:name=".AidlService">
    <intent-filter>
        <action android:name="android.intent.action.AIDLService" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</service>
```

2——客户端编写

Step 1：把服务端的AIDL文件拷贝下，拷贝后目录如下：



Step 2：编写简单的布局,再接着就是核心MainActvitiy的实现了 定义一个ServciceConnection对象,重写对应方法,和前面的普通数据的类似 再接着在bindService,然后再Button的点击事件中获取Salary对象并显示出来！

MainActivity.java

```
package com.jay.example.aidl_complexclient;

import com.jay.example.aidl.ISalary;
import com.jay.example.aidl.Person;
import com.jay.example.aidl.Salary;

import android.app.Activity;
import android.app.Service;
import android.content.ComponentName;
import android.content.Intent;
```

```
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends Activity {

    private ISalary salaryService;
    private Button btnquery;
    private EditText editname;
    private TextView textshow;
    private ServiceConnection conn = new ServiceConnection() {

        @Override
        public void onServiceDisconnected(ComponentName name) {
            salaryService = null;
        }

        @Override
        public void onServiceConnected(ComponentName name, IBinder
            //返回的是代理对象,要调用这个方法哦!
            salaryService = ISalary.Stub.asInterface(service);
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btnquery = (Button) findViewById(R.id.btnquery);
        editname = (EditText) findViewById(R.id.editname);
        textshow = (TextView) findViewById(R.id.textshow);

        Intent it = new Intent();
        it.setAction("com.jay.aidl.AIDL_SERVICE");
        bindService(it, conn, Service.BIND_AUTO_CREATE);

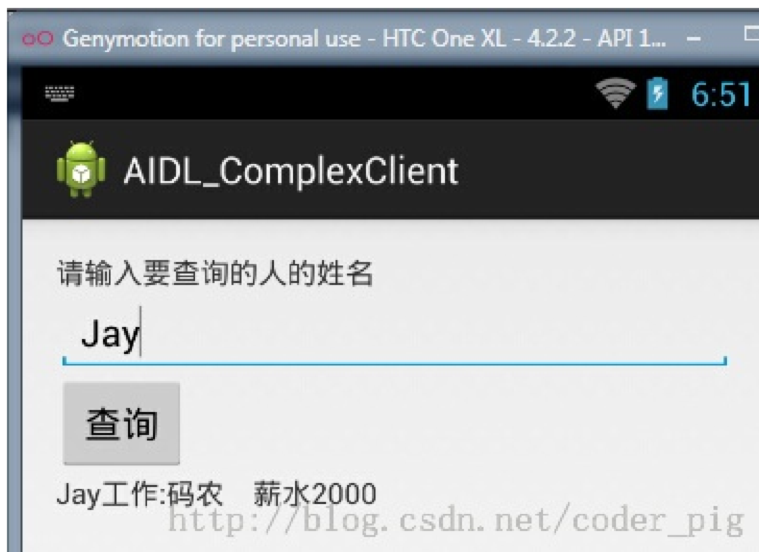
        btnquery.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                try
                {
                    String name = editname.getText().toString();
                    Salary salary = salaryService.getMsg(new Persor
                    textshow.setText(name + salary.toString());
                }catch(RemoteException e){e.printStackTrace();}
            }
```

```
        });

    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        this.unbindService(conn);
    }

}
```

运行截图：



PS： 这里的代码是之前用Eclipse写的代码，Android Studio下自定义类型有点问题， 暂时没找到解决方法，如果知道的朋友请告知下！！！万分感激！！！ 出现的问题如下：

```
Execution failed for task ':app:compileDebugAidl'.
> com.android.ide.common.process.ProcessException: org.gradle.process.internal.ExecException: Process 'command 'C:\Software\Coding\android-sdks-as\build-tools\23.0.0-preview\aidl.exe'' finished with non-zero exit value 1
```

两个实例的代码下载(基于Eclipse的)： 1)使用AIDL完成进程间的简单通信 2） 传递复杂数据的AIDL Service的实现

3.直接通过Binder的onTransact完成跨进程通信

上面讲过Android可以通过Binder的onTrensact方法来完成通信，下面就来简单试下下，还是前面那个根据 序号查询名字的例子：

服务端实现：

```
/**
 * Created by Jay on 2015/8/18 0018.
 */
public class IPCService extends Service{

    private static final String DESCRIPTOR = "IPCService";
    private final String[] names = {"B神","艹神","基神","J神","翔神"}
    private MyBinder mBinder = new MyBinder();

    private class MyBinder extends Binder {
        @Override
        protected boolean onTransact(int code, Parcel data, Parcel
            switch (code){
                case 0x001: {
                    data.enforceInterface(DESCRIPTOR);
                    int num = data.readInt();
                    reply.writeNoException();
                    reply.writeString(names[num]);
                    return true;
                }
            }
            return super.onTransact(code, data, reply, flags);
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

客户端实现：

```
public class MainActivity extends AppCompatActivity implements View

    private EditText edit_num;
    private Button btn_query;
    private TextView txt_result;
    private IBinder mIBinder;
    private ServiceConnection PersonConnection  = new ServiceConnec
    {
        @Override
        public void onServiceDisconnected(ComponentName name)
        {
            mIBinder = null;
        }

        @Override
        public void onServiceConnected(ComponentName name, IBinder
        {
```

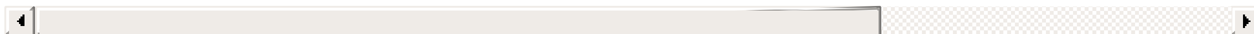
```
            mIBinder = service;
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();

        //绑定远程Service
        Intent service = new Intent("android.intent.action.IPCServi
        service.setPackage("com.jay.ipcserver");
        bindService(service, PersonConnection, BIND_AUTO_CREATE);
        btn_query.setOnClickListener(this);
    }

    private void bindViews() {
        edit_num = (EditText) findViewById(R.id.edit_num);
        btn_query = (Button) findViewById(R.id.btn_query);
        txt_result = (TextView) findViewById(R.id.txt_result);
    }

    @Override
    public void onClick(View v) {
        int num = Integer.parseInt(edit_num.getText().toString());
        if (mIBinder == null)
        {
            Toast.makeText(this, "未连接服务端或服务端被异常杀死", Toast
        } else {
            android.os.Parcel _data = android.os.Parcel.obtain();
            android.os.Parcel _reply = android.os.Parcel.obtain();
            String _result = null;
            try{
                _data.writeInterfaceToken("IPCService");
                _data.writeInt(num);
                mIBinder.transact(0x001, _data, _reply, 0);
                _reply.readException();
                _result = _reply.readString();
                txt_result.setText(_result);
                edit_num.setText("");
            }catch (RemoteException e)
            {
                e.printStackTrace();
            } finally
            {
                _reply.recycle();
                _data.recycle();
            }
        }
    }
}
```



运行截图：



代码比较简单，就不多解释了~用到自己改改即可！ **PS**:代码参考于:Android aidl Binder框架浅析

4.Android 5.0后Service一些要注意的地方：

> 今天在隐式启动Service的时候，遇到这样一个问题
> `java.lang.IllegalArgumentException: Service Intent must be explicit: Intent { act=android.intent.action.AIDLService }`
> 然后程序一启动就崩了,后来苦扣良久才发下是Android 5.0惹的祸， 原来5.0后有个新的特性，就是： **Service Intent must be explitict**！ 好吧，就是不能隐式去启动Service咯，解决的方法也很简单！ 比如StartService的：
>
> **startService(new Intent(getApplicationContext(), "com.aaa.xxxserver"));** 这样写程序直接crash掉，要写成下面这样： **startService(new Intent(getApplicationContext(), LoadContactsService.class));**
>
> 如果是BindService的： **Intent service = new Intent("android.intent.action.AIDLService");** 的基础上，要加上包名： **service.setPackage("com.jay.ipcserver");** 这样就可以了~

官方文档：http://developer.android.com/intl/zh-cn/guide/components/intents-filters.html#Types 文档说明处:

> **Caution:** To ensure your app is secure, always use an explicit intent when starting a `Service` and do not declare intent filters for your services. Using an implicit intent to start a service is a security hazard because you cannot be certain what service will respond to the intent, and the user cannot see which service starts. Beginning with Android 5.0 (API level 21), the system throws an exception if you call `bindService()` with an implicit intent.

本节小结：

> 好的，关于Service的最后一节就到这里，本节讲解了Binder的基本概念以及实现进程间通信的 两种方式：通过AIDL以及Binder.onTransact()来实现跨进程通信！最后还讲解了下Android 5.0后 使用Service不能隐式启动的注意事项！就到这里，谢谢~

4.3.1 BroadcastReceiver牛刀小试

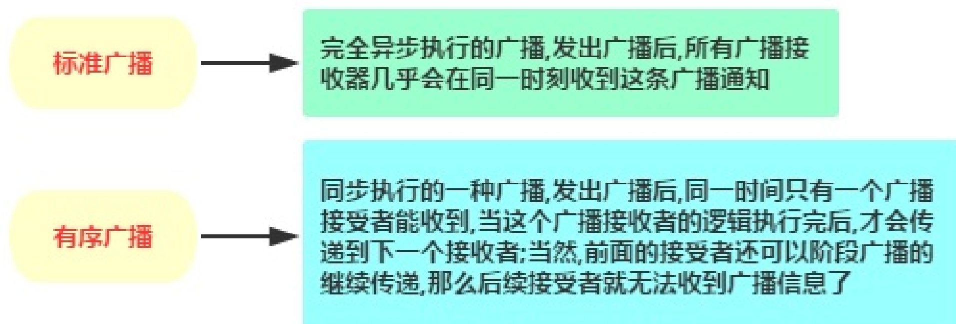
本节引言

本节我们将来学习Android四大组件中的第三个：BroadcastReceiver(广播接收者)，嘿嘿，刚一直在想 如何写开头语，于是乎翻了手头的两本Android基础书，发现两本书都没有对BroadcastReceiver的介绍， 不知道是巧合还是作者觉得这东西用得不多，没必要讲！不过，他们不讲，小猪却会讲，还要详细讲咧！ 好的，开始本节内容~ PS:对了，在Android官网上，点开API Guides -> App Components也没发现有BroadcastReceiver的踪迹， 恩呢，那就直接搜BroadcastReceiver，对应文档地址：BroadcastReceiver

1.BroadcastReceiver是什么鬼？

> 答：Broadcast直译广播，我们举个形象的例子来帮我理解下BroadcastReceiver，记得以前读书 的时候，每个班级都会有一个挂在墙上的大喇叭，用来广播一些通知，比如，开学要去搬书，广播： "每个班级找几个同学教务处拿书"，发出这个广播后，所有同学都会在同一时刻收到这条广播通知， 收到，但不是每个同学都会去搬书，一般去搬书的都是班里的"大力士"，这群"大力士"接到这条 广播后就会动身去把书搬回可是！ ——好吧，上面这个就是一个广播传递的一个很形象的例子： 大喇叭--> 发送广播 --> 所有学生都能收到广播 --> 大力士处理广播 回到我们的概念，其实BroadcastReceiver就是应用程序间的全局大喇叭，即通信的一个手段， 系统自己在很多时候都会发送广播，比如电量低或者充足，刚启动完，插入耳机，输入法改变等， 发生这些时间，系统都会发送广播，这个叫系统广播，每个APP都会收到，如果你想让你的应用在接收到 这个广播的时候做一些操作，比如：系统开机后，偷偷后台跑服务~哈哈，这个时候你只需要为你的应用 注册一个用于监视开机的BroadcastReceiver，当接收到开机广播就做写偷偷摸摸的勾当~ 当然我们也可以自己发广播，比如：接到服务端推送信息，用户在别处登录，然后应该强制用户下线回到 登陆界面，并提示在别处登录~当然，这些等下都会写一个简单的示例帮大家了解广播给我们带来的好处~

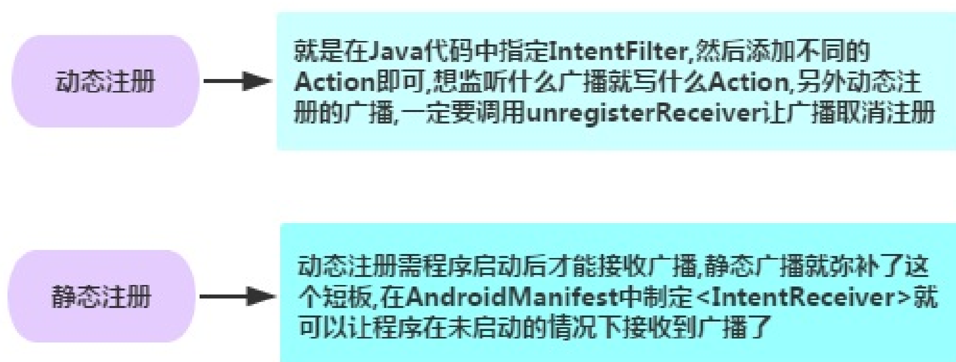
2.两种广播类型：



3.接收系统广播

1）两种注册广播的方式

> 前面也讲了，系统在某些时候会发送相应的系统广播，下面我们就来让我们的APP接收系统广播，接收之前，还需要为我们的APP注册广播接收器哦！而注册的方法又分为以下两种：动态与静态！

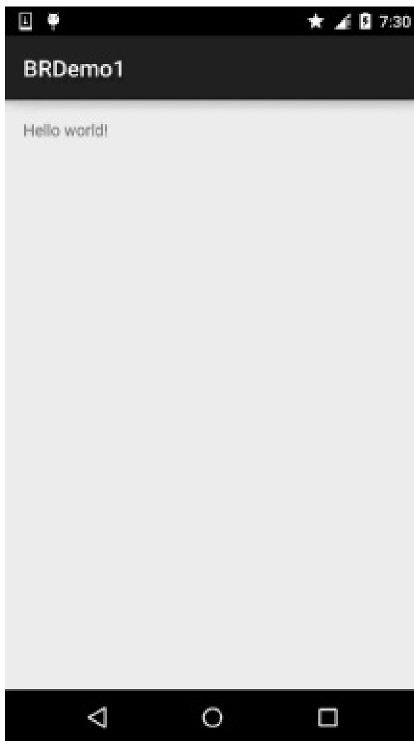


下面我们分别通过代码来演示两者的用法以及不同之处：

2）动态注册实例(监听网络状态变化)

代码示例：

效果图：



好的，一开始是没有联网的，即没有打开wifi，点击打开wifi过了一会儿就出现Toast提示了~ 实现起来也很简单！

代码实现：

自定义一个BroadcastReceiver，在onReceive()方法中完成广播要处理的事务，比如这里的提示Toast信息： MyBRReceiver.java

```
public class MyBRReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context,"网络状态发生改变~",Toast.LENGTH_SHORT
    }
}
```

MainActivity.java中动态注册广播：

```
public class MainActivity extends AppCompatActivity {

    MyBRReceiver myReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //核心部分代码：
        myReceiver = new MyBRReceiver();
        IntentFilter itFilter = new IntentFilter();
        itFilter.addAction("android.net.conn.CONNECTIVITY_CHANGE");
        registerReceiver(myReceiver, itFilter);
    }

    //别忘了将广播取消掉哦~
    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(myReceiver);
    }
}
```

动态注册简单吧~但是动态注册有个缺点就是需要程序启动才可以接收广播，假如我们需要程序 没有启动，但是还是能接收广播的话，那么就需要注册静态广播了！

3）静态注册实例(接收开机广播)

代码示例：

这里就没有示意图了~，直接看代码实现吧~

代码实现：

1.自定义一个**BroadcastReceiver**，重写**onReceive**完成事务处理

```
public class BootCompleteReceiver extends BroadcastReceiver {
    private final String ACTION_BOOT = "android.intent.action.BOOT_
    @Override
    public void onReceive(Context context, Intent intent) {
    if (ACTION_BOOT.equals(intent.getAction()))
        Toast.makeText(context, "开机完毕~", Toast.LENGTH_LONG).show
    }
}
```

2.在**AndroidManifest.xml**中对该**BroadcastReceiver**进行注册，添加开机广播的**intent-filter!**

对了，别忘了加上**android.permission.RECEIVE_BOOT_COMPLETED**的权限哦！

```
<receiver android:name=".BootCompleteReceiver">
    <intent-filter>
        <action android:name = "android.intent.cation.BOOT_COMPLETE
    </intent-filter>
</receiver>

<!-- 权限 -->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMP
```

好的，然后你重启下手机会发现过了一会儿，就会弹出开机完毕这个Toast的了~ 另外，Android 4.3以上的版本，是允许将程序安装到SD卡上的，假如你的程序是安装在SD上 的，就会收不到开机广播，具体原因以及解决方法下一节再详细讲解！

4)使用广播的注意事项：

嘿嘿，广播好用吧，又简单，但是使用广播要注意：

> 不要在广播里添加过多逻辑或者进行任何耗时操作,因为在广播中是不允许开辟线程的, 当onReceiver( )方法运行较长时间(超过10秒)还没有结束的话,那么程序会报错(ANR), 广播更多的时候扮演的是一个打开其他组件的角色,比如启动Service,Notification提示, Activity等！

4.发送广播

> 嗯，上面我们都是接收系统的广播，系统发我们收，我们不能老这么被动，总得主动点是吧！ 另外，明天七夕，程序猿们好好把握，争取脱单，哈哈！好的，说回广播，我们自己主动发广播！ 下面我们就来看下怎么实现！

如何发送： 发送广播前，要先定义一个接收器，先确定目标，然后再告白！~ (●'◡'●)~

发送广播前需要定义一个广播接收器,不然发了谁收...于是乎我们
自定义一个BroadcastReceiver,重写onReceive( )方法,注册下!
①标准广播:sendBroadcast(intent);即可
②有序广播:sendOrderedBroadcast(intent,null);
可以在清单文件中的Intent-filter通过:android:priority="100"设置优先级,值越大
优先级越高,越先收到广播,而且还可以调用abortBroadcast( )截断广播的继续传递
优先级可选值：-1000~1000之间

代码示例：(标准广播)

MyBroadcastReceiver.java

```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private final String ACTION_BOOT = "com.example.broadcasttest.M
    @Override
    public void onReceive(Context context, Intent intent) {
        if(ACTION_BOOT.equals(intent.getAction()))
        Toast.makeText(context, "收到告白啦~",Toast.LENGTH_SHORT).sh
    }
}
```

然后**AndroidManifest.xml**中注册下，写上**Intent-filter**：

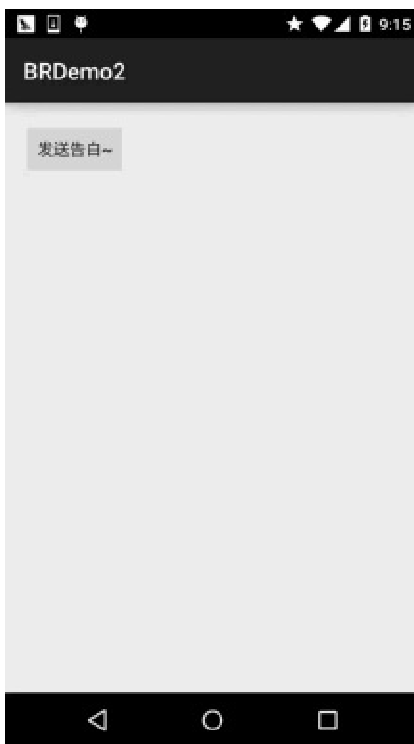
```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="com.example.broadcasttest.MY_BROADCAS
    </intent-filter>
</receiver>
```

好的，接下来我们把上面这个程序项目运行下，然后关掉，接下来我们新建一个项目， 在这个项目里完成广播发送~新建Demo2，布局就一个简单按钮，然后在MainActivity中完成广播发送：

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn_send = (Button) findViewById(R.id.btn_send);
        btn_send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sendBroadcast(new Intent("com.example.broadcastest
            }
        });
    }
}
```

嘿嘿，看下运行截图：



本节小结：

> 好的，BroadcastReceiver的简单使用就是那么简单，不过我们这里用到的都是全局广播，也就是其他 应用也能收到我们的广播，这样可能会引起一些安全性问题，不过没事，下一节我们来教大家如何用 本地广播，以及Android 4.3后如何应用安装到SD卡上，如何监听开机启动~好的，本节就到这里，谢谢~

4.3.2 BroadcastReceiver庖丁解牛

本节引言：

> 上节我们对BroadcastReceiver已经有了一个初步的了解了，知道两种广播类型：标准与有序， 动态或静态注册广播接收者，监听系统广播，自己发送广播！已经满足我们的基本需求了~ 但是前面写的广播都是全局广播！这同样意味着我们APP发出的广播，其他APP都会接收到， 或者其他APP发送的广播，我们的APP也同样会接收到，这样容易引起一些安全性的问题！而 Android中给我们提供了本地广播的机制，使用该机制发出的广播只会在APP内部传播，而且 广播接收者也只能收到本应用发出的广播！

1.本地广播

1）核心用法：

使用LocalBroadcastManager来管理广播:
①调用LocalBroadcastManager.getInstance( )获得实例
②调用~.registerReceiver()注册广播
③调用~.sendBroadcast()发送广播
④调用~.unregisterReceiver()取消注册
PS:~代表LocalBroadcastManager的实例

PS：本地广播无法通过静态注册方式来接受,相比起系统全局广播更加高效

2）注意事项：

1.本地广播无法通过静态注册来接收！相比起系统全局广播更加高效
2.在广播中启动Activity的话,需要为intent加入
FLAG_ACTIVITY_NEW_TASK的标记,不然会报错,因为需要一个栈来存放
新打开的Activity
3.广播中弹出AlertDialog的话,需要设置对话框的类型为
TYPE_SYSTEM_ALERT不然是无法弹出的~

3）代码示例(别处登陆踢用户下线)：

> 像微信一样，正在运行的微信，如果我们用别的手机再次登陆自己的账号，前面这个是会提醒账户 在别的终端登录这样，然后把我们打开的所有Activity都关掉，然后回到登陆页面这样~ 下面我们就来写个简单的例子：

运行效果图：



代码实现：

Step 1：准备一个关闭所有Activity的ActivityCollector，这里之前用前面Activity提供的那个!

ActivityCollector.java

```
public class ActivityCollector {
    private static List<Activity> activities = new ArrayList<Activi
    public static void addActivity(Activity activity) {
        activities.add(activity);
    }
    public static void removeActivity(Activity activity) {
        activities.remove(activity);
    }
    public static void finishAll() {
        for (Activity activity : activities) {
            if (!activity.isFinishing()) {
                activity.finish();
            }
        }
    }
}
```

Step 2：先写要给简单的BaseActivity，用来继承，接着写下登陆界面！

```
public class BaseActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityCollector.addActivity(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        ActivityCollector.removeActivity(this);
    }
}
```

LoginActivity.java:

```
public class LoginActivity extends BaseActivity implements View.On(

    private SharedPreferences pref;
    private SharedPreferences.Editor editor;

    private EditText edit_user;
    private EditText edit_pawd;
    private Button btn_login;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        pref = PreferenceManager.getDefaultSharedPreferences(this);

        bindViews();
    }

    private void bindViews() {
        edit_user = (EditText) findViewById(R.id.edit_user);
        edit_pawd = (EditText) findViewById(R.id.edit_pawd);
        btn_login = (Button) findViewById(R.id.btn_login);
        btn_login.setOnClickListener(this);
    }

    @Override
    protected void onStart() {
        super.onStart();
        if(!pref.getString("user","").equals("")){
            edit_user.setText(pref.getString("user",""));
            edit_pawd.setText(pref.getString("pawd",""));
        }
    }

    @Override
```

```
    public void onClick(View v) {
        String user = edit_user.getText().toString();
        String pawd = edit_pawd.getText().toString();
        if(user.equals("123")&&pawd.equals("123")){
            editor = pref.edit();
            editor.putString("user", user);
            editor.putString("pawd", pawd);
            editor.commit();
            Intent intent = new Intent(LoginActivity.this, MainActi
            startActivity(intent);
            Toast.makeText(LoginActivity.this,"哟，竟然蒙对了~",Toast
            finish();
        }else{
            Toast.makeText(LoginActivity.this,"这么简单都输出，脑子呢？
        }

    }
}
```

Step 3：自定义一个BroadcastReceiver，在onReceive里完成弹出对话框操作，以及启动登陆页面： **MyBcReceiver.java**

```
public class MyBcReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(final Context context, Intent intent) {
        AlertDialog.Builder dialogBuilder = new AlertDialog.Builder
        dialogBuilder.setTitle("警告：");
        dialogBuilder.setMessage("您的账号在别处登录，请重新登陆~");
        dialogBuilder.setCancelable(false);
        dialogBuilder.setPositiveButton("确定",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
                        ActivityCollector.finishAll();
                        Intent intent = new Intent(context, LoginAc
                        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TA
                        context.startActivity(intent);
                    }
                });
        AlertDialog alertDialog = dialogBuilder.create();
        alertDialog.getWindow().setType(
                WindowManager.LayoutParams.TYPE_SYSTEM_ALERT);
        alertDialog.show();
    }
}
```

别忘了AndroidManifest.xml中加上系统对话框权限： **< uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />**

Step 4：在MainActivity中，实例化localBroadcastManager，拿他完成相关操作，另外销毁时 注意unregisterReceiver！

MainActivity.java

```
public class MainActivity extends BaseActivity {

    private MyBcReceiver localReceiver;
    private LocalBroadcastManager localBroadcastManager;
    private IntentFilter intentFilter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        localBroadcastManager = LocalBroadcastManager.getInstance(t

        //初始化广播接收者，设置过滤器
        localReceiver = new MyBcReceiver();
        intentFilter = new IntentFilter();
        intentFilter.addAction("com.jay.mybcreceiver.LOGIN_OTHER");
        localBroadcastManager.registerReceiver(localReceiver, inter

        Button btn_send = (Button) findViewById(R.id.btn_send);
        btn_send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent("com.jay.mybcreceiver.L
                localBroadcastManager.sendBroadcast(intent);
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        localBroadcastManager.unregisterReceiver(localReceiver);
    }
}
```

好的，就是这么简单，别忘记注册Activity哦~

2.Android 4.3以上版本监听开机启动广播的问题解决：

在Android 4.3以上的版本，允许我们将应用安装在SD上，我们都知道是系统开机间隔一小段时间后，才装载SD卡的，这样我们的应用就可能监听不到这个广播了！所以我们需要既监听开机广播又监听SD卡挂载广播！

另外，有些手机可能并没有SD卡，所以这两个广播监听我们不能写到同一个Intetn-filter里 而是应该写成两个，配置代码如下：

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED'
    </intent-filter>
    <intent-filter>
        <action android:name="ANDROID.INTENT.ACTION.MEDIA_MOUNTED"/
        <action android:name="ANDROID.INTENT.ACTION.MEDIA_UNMOUNTED
        <data android:scheme="file"/>
    </intent-filter>
</receiver>
```

3.常用的系统广播总结：

最后给大家提供下我们平常可能会用到的一些系统广播吧：

```
Intent.ACTION_AIRPLANE_MODE_CHANGED;
//关闭或打开飞行模式时的广播

<strong>Intent.ACTION_BATTERY_CHANGED;
//充电状态，或者电池的电量发生变化
//电池的充电状态、电荷级别改变，不能通过组建声明接收这个广播，只有通过Context.

<strong>Intent.ACTION_BATTERY_LOW;
//表示电池电量低

<strong>Intent.ACTION_BATTERY_OKAY;
//表示电池电量充足，即从电池电量低变化到饱满时会发出广播

Intent.ACTION_BOOT_COMPLETED;
//在系统启动完成后，这个动作被广播一次（只有一次）。

Intent.ACTION_CAMERA_BUTTON;
//按下照相时的拍照按键(硬件按键)时发出的广播

Intent.ACTION_CLOSE_SYSTEM_DIALOGS;
//当屏幕超时进行锁屏时,当用户按下电源按钮,长按或短按(不管有没跳出话框)，进行锁

Intent.ACTION_CONFIGURATION_CHANGED;
//设备当前设置被改变时发出的广播(包括的改变:界面语言，设备方向，等，请参考Conf

Intent.ACTION_DATE_CHANGED;
//设备日期发生改变时会发出此广播

Intent.ACTION_DEVICE_STORAGE_LOW;
//设备内存不足时发出的广播,此广播只能由系统使用，其它APP不可用？
```

```
Intent.ACTION_DEVICE_STORAGE_OK;
//设备内存从不足到充足时发出的广播,此广播只能由系统使用，其它APP不可用？

Intent.ACTION_DOCK_EVENT;
//
//发出此广播的地方frameworks\base\services\java\com\android\server\Do

Intent.ACTION_EXTERNAL_APPLICATIONS_AVAILABLE;
////移动APP完成之后，发出的广播(移动是指:APP2SD)

Intent.ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE;
//正在移动APP时，发出的广播(移动是指:APP2SD)

Intent.ACTION_GTALK_SERVICE_CONNECTED;
//Gtalk已建立连接时发出的广播

Intent.ACTION_GTALK_SERVICE_DISCONNECTED;
//Gtalk已断开连接时发出的广播

Intent.ACTION_HEADSET_PLUG;
//在耳机口上插入耳机时发出的广播

Intent.ACTION_INPUT_METHOD_CHANGED;
//改变输入法时发出的广播

Intent.ACTION_LOCALE_CHANGED;
//设备当前区域设置已更改时发出的广播

Intent.ACTION_MANAGE_PACKAGE_STORAGE;
//

Intent.ACTION_MEDIA_BAD_REMOVAL;
//未正确移除SD卡(正确移除SD卡的方法:设置--SD卡和设备内存--卸载SD卡)，但已把S
//广播：扩展介质（扩展卡）已经从 SD 卡插槽拔出，但是挂载点 (mount point) 还

Intent.ACTION_MEDIA_BUTTON;
//按下"Media Button" 按键时发出的广播,假如有"Media Button" 按键的话(硬件按

Intent.ACTION_MEDIA_CHECKING;
//插入外部储存装置，比如SD卡时，系统会检验SD卡，此时发出的广播?
Intent.ACTION_MEDIA_EJECT;
//已拔掉外部大容量储存设备发出的广播（比如SD卡，或移动硬盘）,不管有没有正确卸载
//广播：用户想要移除扩展介质（拔掉扩展卡）。
Intent.ACTION_MEDIA_MOUNTED;
//插入SD卡并且已正确安装（识别）时发出的广播
//广播：扩展介质被插入，而且已经被挂载。
Intent.ACTION_MEDIA_NOFS;
//
Intent.ACTION_MEDIA_REMOVED;
//外部储存设备已被移除，不管有没正确卸载,都会发出此广播？
// 广播：扩展介质被移除。
Intent.ACTION_MEDIA_SCANNER_FINISHED;
```

```
//广播：已经扫描完介质的一个目录
Intent.ACTION_MEDIA_SCANNER_SCAN_FILE;
//
Intent.ACTION_MEDIA_SCANNER_STARTED;
//广播：开始扫描介质的一个目录

Intent.ACTION_MEDIA_SHARED;
// 广播：扩展介质的挂载被解除 (unmount)，因为它已经作为 USB 大容量存储被共享
 Intent.ACTION_MEDIA_UNMOUNTABLE;
//
Intent.ACTION_MEDIA_UNMOUNTED
// 广播：扩展介质存在，但是还没有被挂载 (mount)。
Intent.ACTION_NEW_OUTGOING_CALL;

Intent.ACTION_PACKAGE_ADDED;
//成功的安装APK之后
//广播：设备上新安装了一个应用程序包。
//一个新应用包已经安装在设备上，数据包括包名（最新安装的包程序不能接收到这个广播
 Intent.ACTION_PACKAGE_CHANGED;
//一个已存在的应用程序包已经改变，包括包名
Intent.ACTION_PACKAGE_DATA_CLEARED;
//清除一个应用程序的数据时发出的广播(在设置－－应用管理－－选中某个应用，之后点
//用户已经清除一个包的数据，包括包名（清除包程序不能接收到这个广播）

Intent.ACTION_PACKAGE_INSTALL;
//触发一个下载并且完成安装时发出的广播，比如在电子市场里下载应用？
//
Intent.ACTION_PACKAGE_REMOVED;
//成功的删除某个APK之后发出的广播
//一个已存在的应用程序包已经从设备上移除，包括包名（正在被安装的包程序不能接收到

Intent.ACTION_PACKAGE_REPLACED;
//替换一个现有的安装包时发出的广播（不管现在安装的APP比之前的新还是旧，都会发出
Intent.ACTION_PACKAGE_RESTARTED;
//用户重新开始一个包，包的所有进程将被杀死，所有与其联系的运行时间状态应该被移除
Intent.ACTION_POWER_CONNECTED;
//插上外部电源时发出的广播
Intent.ACTION_POWER_DISCONNECTED;
//已断开外部电源连接时发出的广播
Intent.ACTION_PROVIDER_CHANGED;
//

Intent.ACTION_REBOOT;
//重启设备时的广播

Intent.ACTION_SCREEN_OFF;
//屏幕被关闭之后的广播

Intent.ACTION_SCREEN_ON;
//屏幕被打开之后的广播

Intent.ACTION_SHUTDOWN;
//关闭系统时发出的广播
```

```
Intent.ACTION_TIMEZONE_CHANGED;
//时区发生改变时发出的广播

Intent.ACTION_TIME_CHANGED;
//时间被设置时发出的广播

Intent.ACTION_TIME_TICK;
//广播：当前时间已经变化（正常的时间流逝）。
//当前时间改变，每分钟都发送，不能通过组件声明来接收，只有通过Context.registe

Intent.ACTION_UID_REMOVED;
//一个用户ID已经从系统中移除发出的广播
//

Intent.ACTION_UMS_CONNECTED;
//设备已进入USB大容量储存状态时发出的广播？

Intent.ACTION_UMS_DISCONNECTED;
//设备已从USB大容量储存状态转为正常状态时发出的广播？

Intent.ACTION_USER_PRESENT;
//

Intent.ACTION_WALLPAPER_CHANGED;
//设备墙纸已改变时发出的广播
```

4.本节小结：

好的，关于BroadcastReceiver的学习就到这里，如果你有什么补充或者建议，欢迎提出~ 万分感激~

<split>123</split>

4.4.1 ContentProvider初探

本节引言：

> 本节给大家带来的是Android四大组件中的最后一个——ContentProvider(内容提供者)，可能部分读者 有疑问了，"Android不是有五大组件的吗？还有个Intent呢？"对的，Intent也是很重要的，但是他 只是维系这几个组件间的纽带！Intent我们下一章会讲解！说会这个ContentProvider，我们什么时候 会用到他呢？有下面这两种：
>
> - **1.**我们想在自己的应用中访问别的应用，或者说一些ContentProvider暴露给我们的一些数据， 比如手机联系人，短信等！我们想对这些数据进行读取或者修改，这就需要用到ContentProvider了！
> - **2.**我们自己的应用，想把自己的一些数据暴露出来，给其他的应用进行读取或操作，我们也可以用 到ContentProvider，另外我们可以选择要暴露的数据，就避免了我们隐私数据的的泄露！
>
> 好像好流弊的样子，其实用起来也很简单，下面我们来对ContentProvider进行学习~ 官方文档：ContentProvider 本节我们来讲解下ContentProvder的概念，给大家写几个常用的使用系统ContentProvider的示例， 以及自定义ContentProvider！

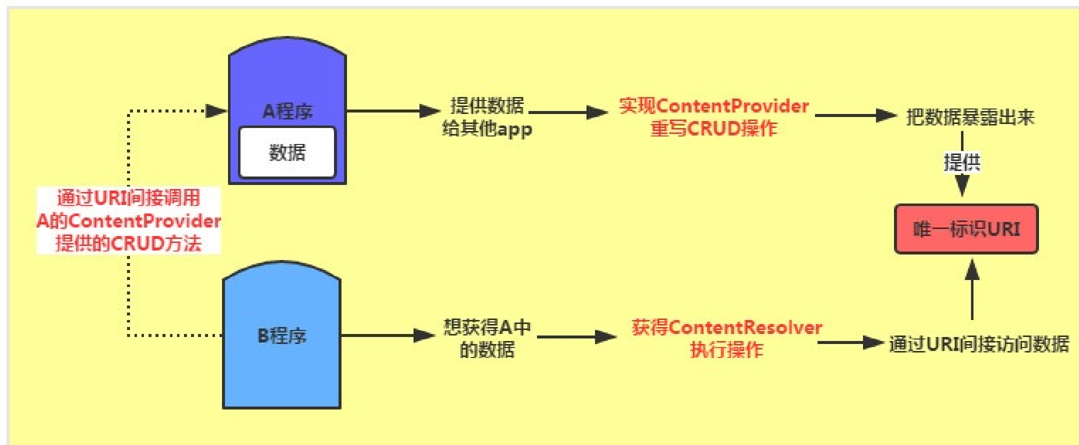
1.ContentProvider概念讲解：

ContentProvider(内容提供者)

ContentProvider的概述:

当我们想允许自己的应用的数据允许别的应用进行读取操作,我们可以让我们的App实现ContentProvider类,同时注册一个Uri,
然后其他应用只要使用ContentResolver根据Uri就可以操作我们的app中的数据了！而数据不一定是数据库,也可能是文件,xml
或者其他,但是SharedPreference使用基于数据库模型的简单表格来提供其中的数据！

ContentProvider的执行原理



URI简介:

专业名词叫做:通用资源标识符,而你也可以类比为网页的域名,我们暂且就把他叫做资源定位符吧,就是定位资源所在路径的
而在本节ContentProvider中,Uri灰常重要,我们分析一个简单的例子吧:
content://com.jay.example.providers.myprovider/word/2

分析:

content:协议头,这个是规定的,就像http,ftp等一样,规定的,而ContentProvider规定的是content开头的
接着是provider所在的全限定类名
word:代表资源部分,如果想访问word所有资源,后面的2就不用写了,直接写word
2:访问的是word资源中id为2的记录

附加

当然,上面也说过数据不仅仅来自于数据库,有时也来源于文件,xml或者网络等其他存储方式,
但是依旧可以使用上面这种URI定义方式:
比如：当表示的xml文件时:~/word/detail表示word节点下的detail结点
另外:URI还提供一个parse()方法将字符串转换为URI
eg:Uri uri = Uri.parese("Content://~");

2.使用系统提供的ContentProvider

> 其实很多时候我们用到ContentProvider并不是自己暴露自己的数据，更多的时候通过 **ContentResolver**来读取其他应用的信息，最常用的莫过于读取系统APP，信息，联系人， 多媒体信息等！如果你想来调用这些ContentProvider就需要自行查阅相关的API资料了！ 另外，不同的版本，可能对应着不同的URL！这里给出如何获取URL与对应的数据库表的字段， 这里以最常用的联系人为例，其他自行google~ ①来到系统源码文件下:all-src.rar -> TeleponeProvider -> AndroidManifest.xml查找对应API ②打开模拟器的file exploer/data/data/com.android.providers.contacts/databases/contact2.db 导出后使用SQLite图形工具查看，三个核心的表:**raw_contact**表，**data**表，**mimetypes**表！ 下面演示一些基本的操作示例：

1）简单的读取收件箱信息：

核心代码：

```
private void getMsgs(){
    Uri uri = Uri.parse("content://sms/");
    ContentResolver resolver = getContentResolver();
    //获取的是哪些列的信息
    Cursor cursor = resolver.query(uri, new String[]{"address","dat
    while(cursor.moveToNext())
    {
        String address = cursor.getString(0);
        String date = cursor.getString(1);
        String type = cursor.getString(2);
        String body = cursor.getString(3);
        System.out.println("地址:" + address);
        System.out.println("时间:" + date);
        System.out.println("类型:" + type);
        System.out.println("内容:" + body);
        System.out.println("======================");
    }
    cursor.close();
}
```

别忘了，往AndroidManifest.xml加入读取收件箱的权限：

```
<uses-permission android:name="android.permission.READ_SMS"/>
```

运行结果：

部分运行结果如下：

```
=====================
地址:1065866983
时间:1413527884228
类型:1
内容:欢迎登录WLAN，如非本人使用可发CZWLANMM到10086重置密码。温馨提醒：2014年4月30日起，非套餐客户资费按0.1元/MB收取
=====================
地址:1065866983
时间:1413527867346
类型:1
内容:您申请的动态密码为：141296
=====================
地址:10658000
时间:1413517039306
类型:1
内容:新闻早晚报快讯：原铁道部运输局局长张曙光受贿案一审宣判，张曙光被判处死刑，缓期二年执行。详见：http://isjb.cn/IYnEqVV
=====================
```

2）简单的往收件箱里插入一条信息

核心代码：

```
private void insertMsg() {
    ContentResolver resolver = getContentResolver();
    Uri uri = Uri.parse("content://sms/");
    ContentValues conValues = new ContentValues();
    conValues.put("address", "123456789");
    conValues.put("type", 1);
    conValues.put("date", System.currentTimeMillis());
    conValues.put("body", "no zuo no die why you try!");
    resolver.insert(uri, conValues);
    Log.e("HeHe", "短信插入完毕~");
}
```

运行结果：



注意事项：

上述代码在4.4以下都可以实现写入短信的功能，而5.0上就无法写入，原因是：从5.0开始，默认短信应用外的软件不能以写入短信数据库的形式发短信！

3）简单的读取手机联系人

核心代码：

```
private void getContacts(){
    //①查询raw_contacts表获得联系人的id
    ContentResolver resolver = getContentResolver();
    Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
    //查询联系人数据
    cursor = resolver.query(uri, null, null, null, null);
    while(cursor.moveToNext())
    {
        //获取联系人姓名,手机号码
        String cName = cursor.getString(cursor.getColumnIndex(Conta
        String cNum = cursor.getString(cursor.getColumnIndex(Contac
        System.out.println("姓名:" + cName);
        System.out.println("号码:" + cNum);
        System.out.println("======================");
    }
    cursor.close();
}
```

别忘了加读联系人的权限：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

运行结果：

部分运行结果如下：

```
======================
姓名:啊菊
号码:6303 65
======================
姓名:傻机
号码:633229
======================
姓名:爽
号码:633290
======================
```

4）查询指定电话的联系人信息

核心代码：

```
private void queryContact(String number){
        Uri uri = Uri.parse("content://com.android.contacts/data/ph
        ContentResolver resolver = getContentResolver();
        Cursor cursor = resolver.query(uri, new String[]{"display_n
        if (cursor.moveToFirst()) {
            String name = cursor.getString(0);
            System.out.println(number + "对应的联系人名称：" + name);
        }
    cursor.close();
}
```

运行结果：

```
I/System.out： 633290对应的联系人名称：爽
```

5）添加一个新的联系人

核心代码：

```
private void AddContact() throws RemoteException, OperationApplicat
    //使用事务添加联系人
    Uri uri = Uri.parse("content://com.android.contacts/raw_contact
    Uri dataUri =  Uri.parse("content://com.android.contacts/data")

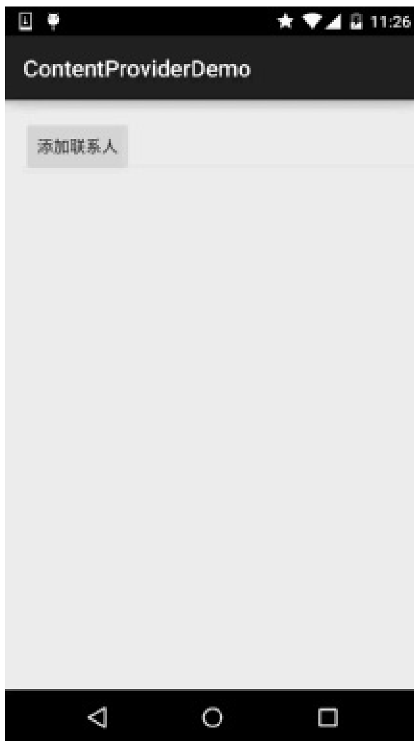
    ContentResolver resolver = getContentResolver();
    ArrayList<ContentProviderOperation> operations = new ArrayList<
    ContentProviderOperation op1 = ContentProviderOperation.newInse
            .withValue("account_name", null)
            .build();
    operations.add(op1);

    //依次是姓名，号码，邮编
    ContentProviderOperation op2 = ContentProviderOperation.newInse
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/name")
            .withValue("data2", "Coder-pig")
            .build();
    operations.add(op2);

    ContentProviderOperation op3 = ContentProviderOperation.newInse
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/phone_v
            .withValue("data1", "13798988888")
            .withValue("data2", "2")
            .build();
    operations.add(op3);

    ContentProviderOperation op4 = ContentProviderOperation.newInse
            .withValueBackReference("raw_contact_id", 0)
            .withValue("mimetype", "vnd.android.cursor.item/email_v
            .withValue("data1", "779878443@qq.com")
            .withValue("data2", "2")
            .build();
    operations.add(op4);
    //将上述内容添加到手机联系人中~
    resolver.applyBatch("com.android.contacts", operations);
    Toast.makeText(getApplicationContext(), "添加成功", Toast.LENGTH
}
```

运行结果:

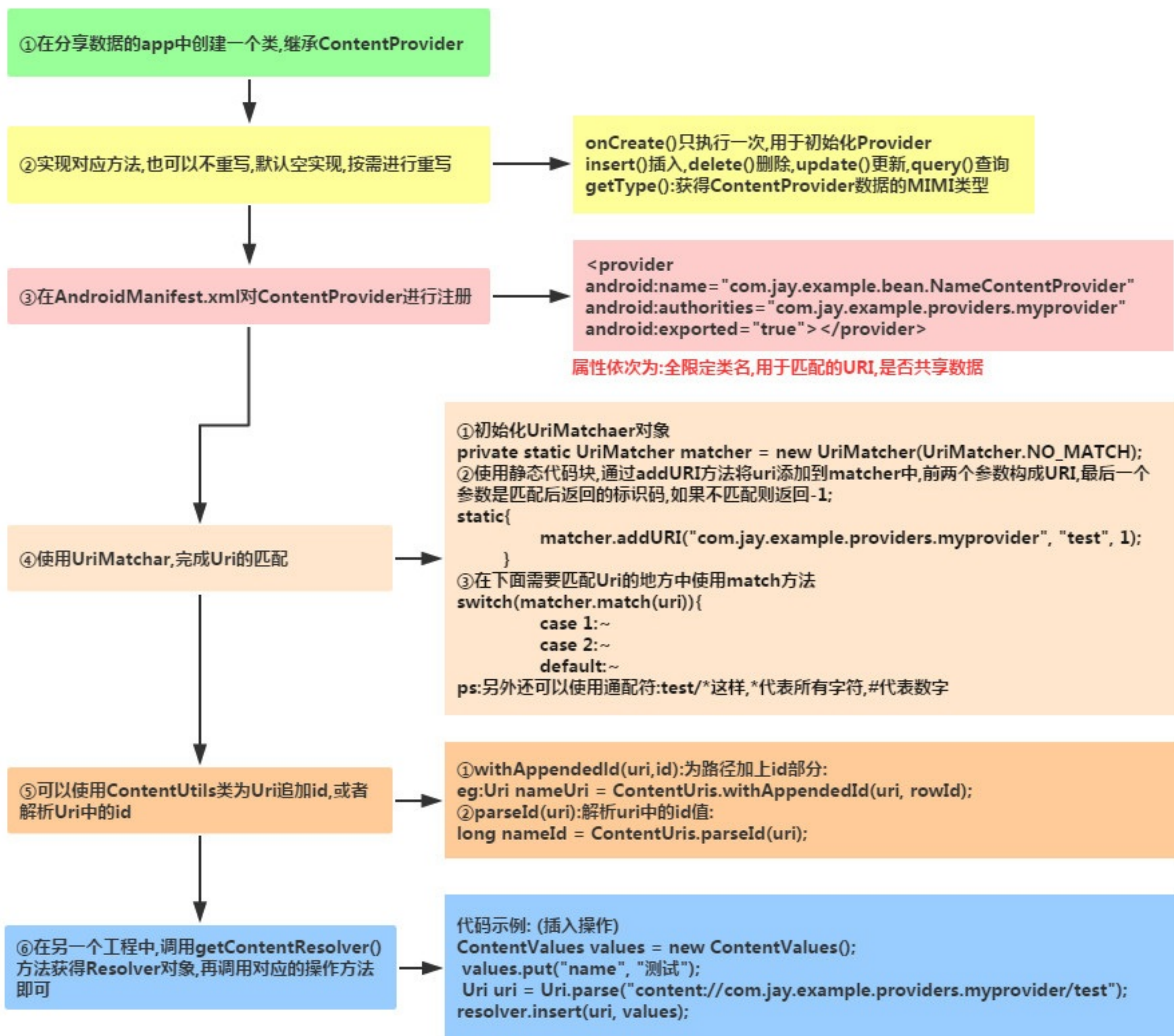


别忘了权限：

```
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_PROFILE"/>
```

3.自定义ContentProvider

> 我们很少会自己来定义ContentProvider，因为我们很多时候都不希望自己应用的数据暴露给 其他应用，虽然这样，学习如何ContentProvider还是有必要的，多一种数据传输的方式，是吧~ 这是之前画的一个流程图：



接下来我们就来一步步实现：

在开始之前我们先要创建一个数据库创建类(数据库内容后面会讲~)：

DBOpenHelper.java

```
public class DBOpenHelper extends SQLiteOpenHelper {

    final String CREATE_SQL = "CREATE TABLE test(_id INTEGER PRIMAR

    public DBOpenHelper(Context context, String name, CursorFactory
            int version) {
        super(context, name, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_SQL);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int ne
        // TODO Auto-generated method stub

    }

}
```

Step 1：自定义ContentProvider类，实现onCreate()，getType()，根据需求重写对应的增删改查方法：

NameContentProvider.java

```
public class NameContentProvider extends ContentProvider {

    //初始化一些常量
     private static UriMatcher matcher = new UriMatcher(UriMatcher.
     private DBOpenHelper dbOpenHelper;

    //为了方便直接使用UriMatcher,这里addURI,下面再调用Matcher进行匹配

     static{
         matcher.addURI("com.jay.example.providers.myprovider", "te
     }

    @Override
    public boolean onCreate() {
        dbOpenHelper = new DBOpenHelper(this.getContext(), "test.db
        return true;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String select
            String[] selectionArgs, String sortOrder) {
        return null;
    }
```

```
    @Override
    public String getType(Uri uri) {
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {

        switch(matcher.match(uri))
        {
        //把数据库打开放到里面是想证明uri匹配完成
        case 1:
            SQLiteDatabase db = dbOpenHelper.getReadableDatabase()
            long rowId = db.insert("test", null, values);
            if(rowId > 0)
            {
                //在前面已有的Uri后面追加ID
                Uri nameUri = ContentUris.withAppendedId(uri, rowI
                //通知数据已经发生改变
                getContext().getContentResolver().notifyChange(name
                return nameUri;
            }
        }
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectio
        return 0;
    }

    @Override
    public int update(Uri uri, ContentValues values, String select
            String[] selectionArgs) {
        return 0;
    }

}
```

Step 2：AndroidManifest.xml中为ContentProvider进行注册：

```
<!--属性依次为：全限定类名,用于匹配的URI,是否共享数据 -->
<provider android:name="com.jay.example.bean.NameContentProvider"
            android:authorities="com.jay.example.providers.myprovi
            android:exported="true" />
```

好的，作为ContentProvider的部分就完成了！

接下来，创建一个新的项目，我们来实现ContentResolver的部分，我们直接通过按钮点击插入一条数据：

MainActivity.java

```
public class MainActivity extends Activity {

    private Button btninsert;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btninsert = (Button) findViewById(R.id.btninsert);

        //读取contentprovider 数据
        final ContentResolver resolver = this.getContentResolver()

        btninsert.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                 ContentValues values = new ContentValues();
                 values.put("name", "测试");
                 Uri uri = Uri.parse("content://com.jay.example.pro
                resolver.insert(uri, values);
                Toast.makeText(getApplicationContext(), "数据插入成功

            }
        });

    }
}
```

如何使用？好吧，代码还是蛮简单的，先运行作为ContentProvider的项目，接着再运行ContentResolver的项目，点击按钮插入一条数据，然后打开file exploer将ContentProvider的db数据库取出，用图形查看工具查看即可发现插入数据，时间关系，就不演示结果了~

4.通过ContentObserver监听ContentProvider的数据变化

使用ContentObserver监听ContentProvider的数据变化

1.自定义类继承ContentObserver → 2.重写onChange方法 → 3.调用getContentResolver().registerContentObserver()注册监听器

代码示例:使用ContentObserver监听用户发送短信

```
package com.jay.example.smsobserver;

import android.app.Activity;
import android.database.ContentObserver;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //①为content://smsm的数据改变注册监听器
        getContentResolver().registerContentObserver(Uri.parse("content://sms"), true,
                new MyObserver(new Handler()));
    }

    private final class MyObserver extends ContentObserver
    {

        public MyObserver(Handler handler) {
            super(handler);
        }
        @Override
        public void onChange(boolean selfChange) {
            //查询发送箱哩的短信(处于正在发送状态的信息放在发送箱)
            Cursor cursor = getContentResolver().query(Uri.parse("content://sms/outbox"),
                    null, null, null, null);
            //遍历查询得到的结果集,就可以获得用户正在发送的短信了
            while(cursor.moveToNext())
            {
                StringBuilder sb = new StringBuilder();
                //发送地址
                sb.append("address=").append(cursor.getString(cursor.getColumnIndex("address")));
                //短信标题
                sb.append(";subject").append(cursor.getString(cursor.getColumnIndex("subject")));
                //短信内容
                sb.append(";body").append(cursor.getString(cursor.getColumnIndex("body")));
                //短信标题
                sb.append(";time").append(cursor.getLong(cursor.getColumnIndex("date")));
                System.out.println("用户发送出去的信息:" + sb.toString());
            }
        }
    }

}
```

使用指南：

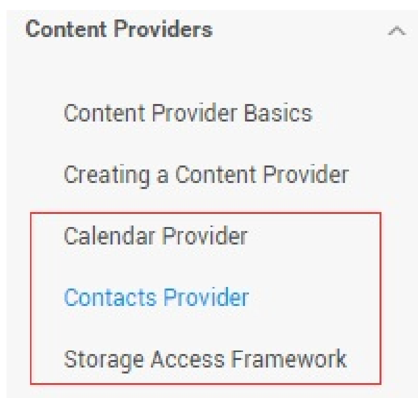
> 运行程序后，晾一边，收到短信后，可以在logcat上看到该条信息的内容，可以根据自己的需求 将Activtiy改做Service，而在后台做这种事情~

本节小结：

> 好的，关于ContentProvider的初探就到这里，本节我们学习了：ContentProvider的概念以及流程，使用系统提供的一些ContentProvider，以及定制自己的ContentProvider， 最后还讲解了通过ContentObserver监听ContentProvider的数据变化，ContentProvider的内容就掌握得差不多 了，下一节我们来走走文档看下有什么不知道的~谢谢

4.4.2 ContentProvider再探——Document Provider

本节引言：

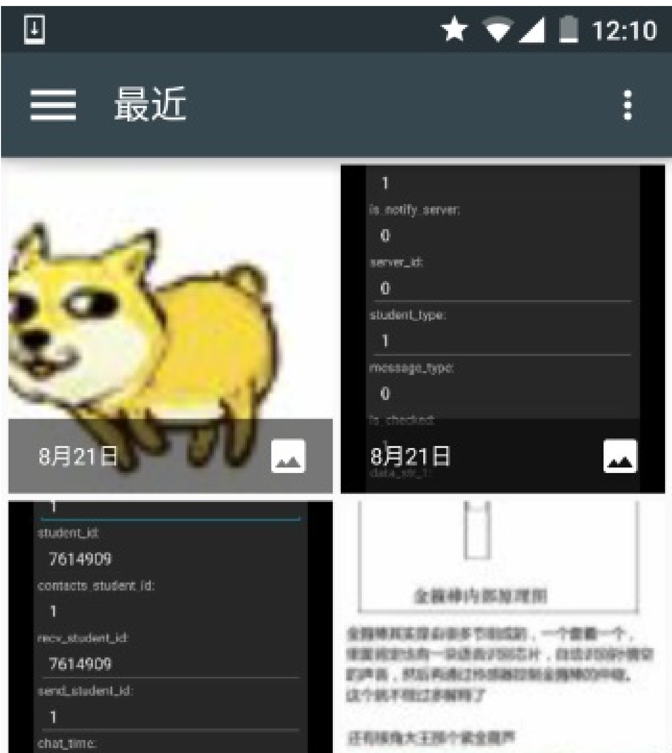
> 学完上一节，相信你已经知道如何去使用系统提供的ContentProvider或者自定义ContentProvider了， 已经基本满足日常开发的需求了，有趣的是，我在官方文档上看到了另外这几个Provider：
>
> 
>
>
> **Calendar Provider**：日历提供者，就是针对针对日历相关事件的一个资源库，通过他提供的API，我们 可以对日历，时间，会议，提醒等内容做一些增删改查！ **Contacts Provider**：联系人提供者，这个就不用说了，这个用得最多~后面有时间再回头翻译下这篇文章吧！ **Storage Access Framework(SAF)**：存储访问框架，4.4以后引入的一个新玩意，为用户浏览手机中的 存储内容提供了便利，可供访问的内容不仅包括：文档，图片，视频，音频，下载，而且包含所有由 由特定ContentProvider（须具有约定的API）提供的内容。不管这些内容来自于哪里，不管是哪个应 用调用浏览系统文件内容的命令，系统都会用一个统一的界面让你去浏览。 其实就是一个内置的应用程序，叫做DocumentsUI，因为它的IntentFilter不带有LAUNCHER，所以我们并没有 在桌面上找到这个东东！嘿嘿，试下下面的代码，这里我们选了两个手机来对比： 分别是4.2的Lenovo S898T 和 5.0.1的Nexus 5做对比，执行下述代码：

```
Intent intent =  new  Intent(Intent.ACTION_OPEN_DOCUMENT); intent
```

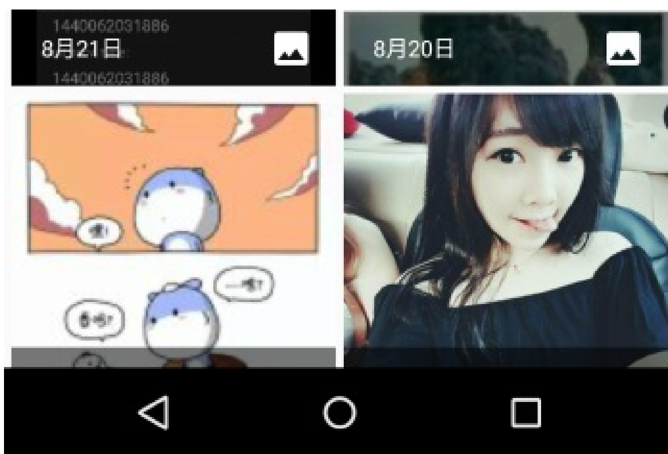
下面是运行结果：



下午 9:52
ContentProviderDemo
打开看图应用
选择要使用的应用：
超级相册
文件管理
WPS Office
始终
仅此一次



12:10
最近
8月21日
8月21日



右面这个就是4.4给我们带来的新玩意了，一般我们获取文件Url的时候就可以用到它~ 接下来简单的走下文档吧~

2.简单走下文档：

1）SAF框架的组成：

> - **Document provider**：一个特殊的ContentProvider，让一个存储服务(比如Google Drive)可以 对外展示自己所管理的文件。它是**DocumentsProvider**的子类， 另外，document-provider的存储格式 和传统的文件存储格式一致，至于你的内容如何存储，则完全决定于你自己，Android系统已经内置了几个 这样的Document provider，比如关于下载，图片以及视频的Document provider！
> - **Client app**：一个普通的客户端软件，通过触发 **ACTION_OPEN_DOCUMENT** 和/或 **ACTION_CREATE_DOCUMENT**就可以接收到来自于Document provider返回的内容，比如选择一个图片，然后返回一个Uri。
> - **Picker**：类似于文件管理器的界面，而且是系统级的界面，提供额访问客户端过滤条件的 Document provider内容的通道，就是起说的那个DocumentsUI程序！

一些特性：

> - 用户可以浏览所有document provider提供的内容，而不仅仅是单一的应用程序
> - 提供了长期、持续的访问document provider中文件的能力以及数据的持久化， 用户可以实现添加、删除、编辑、保存document provider所维护的内容
> - 支持多用户以及临时性的内容服务，比如USB storage providers只有当驱动安装成功才会出现

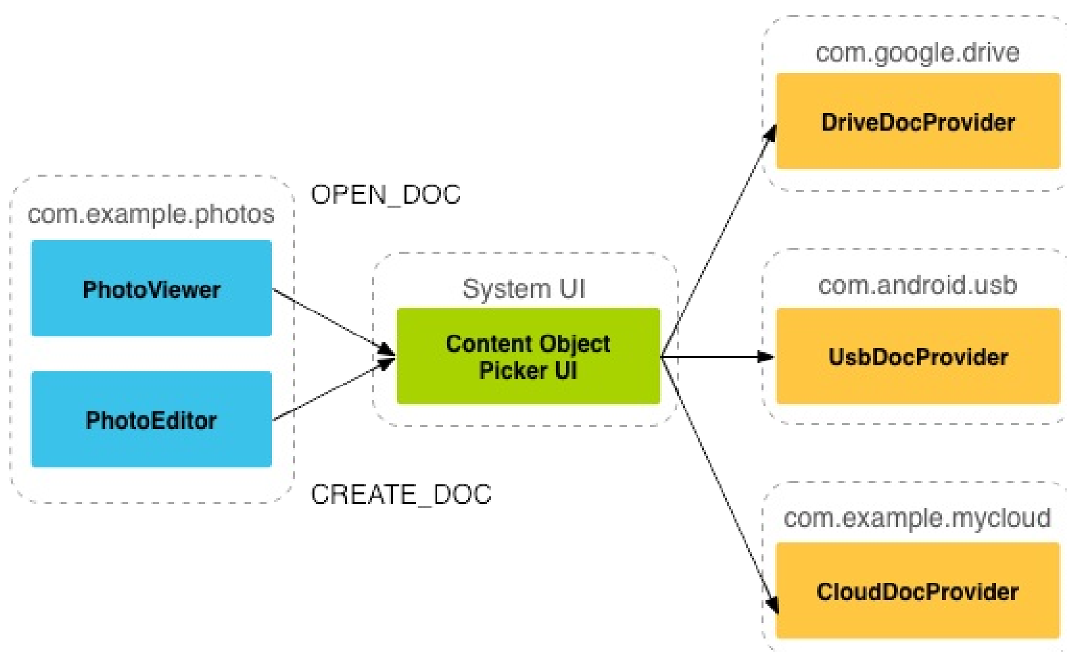
2）概述：

SAF的核心是实现了DocumentsProvider的子类，还是一个ContentProvider。在一个document provider 中是以传统的文件目录树组织起来的：



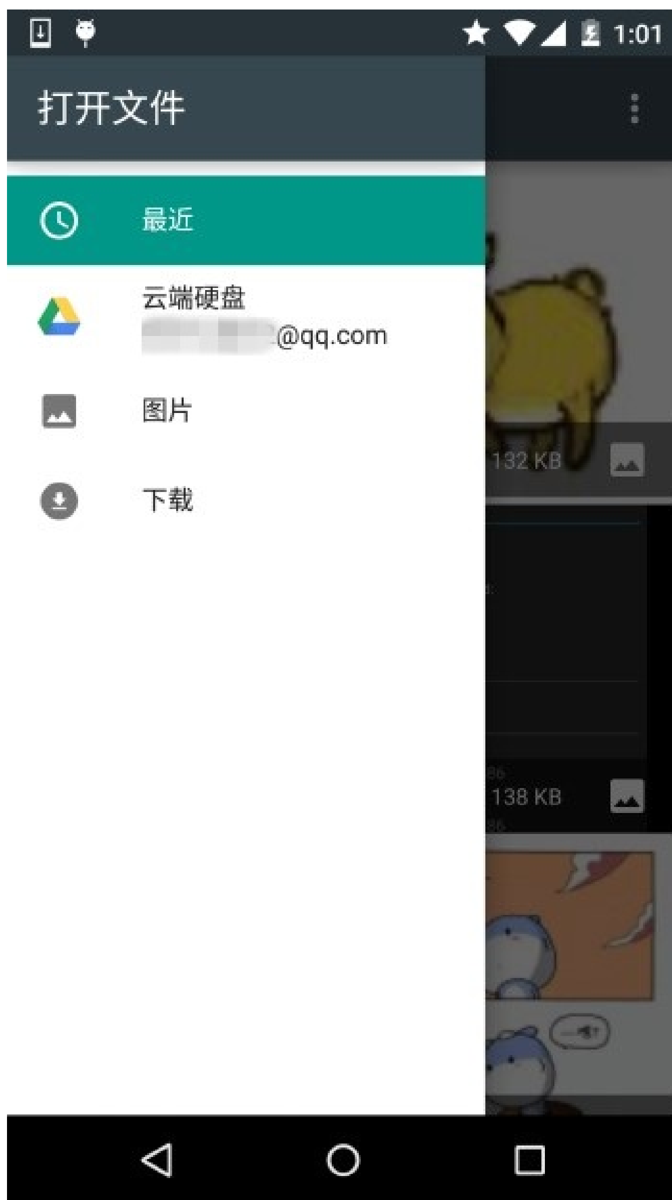
3）流程图：

如上面所述，document provider data是基于传统的文件层次结构的，不过那只是对外的表现形式， 如何存储你的数据，取决于你自己，只要你对海外的接口能够通过DocumentsProvider的api访问就可以。 下面的流程图展示了一个photo应用使用SAF可能的结构：



分析：

> 从上图，我们可以看出Picker是链接调用者和内容提供者的一个桥梁！他提供并告诉调用者，可以选择 哪些内容提供者，比如这里的DriveDocProvider，UsbDocProvider，CloundDocProvider。 当客户端触发了**ACTION_OPEN_DOCUMENT**或**ACTION_CREATE_DOCUMENT**的Intent，就会发生上述交互。 当然我们还可以在Intent中增加过滤条件，比如限制MIME type的类型为"image"!



就是上面这些东西，如果你还安装了其他看图的软件的话，也会在这里看到！ 简单点说就是：客户端发送了上面两种Action的Intent后，会打开Picker UI，在这里会显示相关可用的 Document Provider，供用户选择，用户选择后可以获得文件的相关信息！

4）客户端调用，并获取返回的Uri

实现代码如下：

```
public class MainActivity extends AppCompatActivity implements
```

运行结果：比如我们选中那只狗，然后Picker UI自己会关掉，然后Logcat上可以看到这样一个uri:

```
/com.jay.contentproviderdemo E/HeHe : Uri: content://com.android.providers.media.documents/document/image%3A73072
```

5）根据uri获取文件参数

核心代码如下：

```
public  void dumpImageMetaData(Uri uri)  {  Cursor cursor = getCont
```

运行结果：还是那只狗，调用方法后会输入文件名以及文件大小，以byte为单位

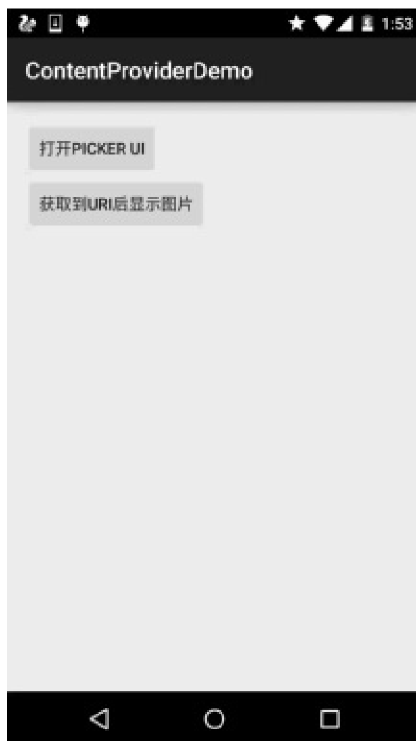
```
Uri: content://com.android.providers.media.documents/document/image%3A73072
Display Name: 779878443D7B744B454CD92631DC38F61C4DA3BB0.jpg
Size: 135338
```

6）根据Uri获得Bitmap

核心代码如下：

```
private  Bitmap getBitmapFromUri(Uri uri)  throws  IOException  {
```

运行结果：



7）根据Uri获取输入流

核心代码如下：

```
private  String readTextFromUri(Uri uri)  throws  IOException  {  I
```

上述的内容只告诉你通过一个Uri你可以知道什么，而Uri的获取则是通过SAF得到的！

8） 创建新文件以及删除文件：

创建文件：

```
private  void createFile(String mimeType,  String fileName)  {  Int
```

可在onActivityResult()中获取被创建文件的uri

删除文件：

前提是Document.COLUMN_FLAGS包含**SUPPORTS_DELETE**

```
DocumentsContract.deleteDocument(getContentResolver(), uri);
```

9）编写一个自定义的Document Provider

如果你希望自己应用的数据也能在documentsui中打开，你就需要写一个自己的document provider。 下面介绍自定义DocumentsProvider的步骤：

> - API版本为19或者更高
> - 在manifest.xml中注册该Provider
> - Provider的name为类名加包名，比如：**com.example.android.storageprovider.MyCloudProvider**
> - Authority为包名+provider的类型名，如：**com.example.android.storageprovider.documents**
> - **android:exported**属性的值为**ture**

下面是Provider的例子写法：

```
<manifest... > ... <uses-sdk  android:minSdkVersion="19"  android:t
```

10)DocumentsProvider的子类

至少实现如下几个方法：

> - queryRoots()
> - queryChildDocuments()
> - queryDocument()
> - openDocument()

还有些其他的方法，但并不是必须的。下面演示一个实现访问文件（file）系统的DocumentsProvider的大致写法。

Implement queryRoots

```
@Override  public  Cursor queryRoots(String[] projection)  throws
```

Implement queryChildDocuments

```
public  Cursor queryChildDocuments(String parentDocumentId,  String
```

Implement queryDocument

```
@Override  public  Cursor queryDocument(String documentId,  String
```

好吧，文档中的内容大概就是这些了： 一开始是想自己翻译的，后来在泡在网上的日子上找到了这一篇文档的中文翻译，就偷下懒了~

中文翻译链接：android存储访问框架Storage Access Framework

3.Android 4.4 获取资源路径问题：

其实这个SAF我们用得较多的地方无非是获取图片的Uri而已，而从上面的例子我们也发现了： 我们这样获取的链接是这样的：

```
content://com.android.providers.media.documents/document/image%3A69
```

这样的链接，我们直接通过上面的方法获得uri即可！

当然，这个是4.4 或者以上版本的~！

如果是以前的版本：uri可能是这样的：

```
content://media/external/images/media/image%3A69983
```

这里贴下在别的地方看到的一个全面的方案，原文链接：Android4.4中获取资源路径问题

```
public  static  String getPath(final  Context context,  final  Uri
 * Get the value of the data column for this Uri. This is useful fo
 * MediaStore Uris, and other file-based ContentProviders.
 *
 * @param context The context.
 * @param uri The Uri to query.
 * @param selection (Optional) Filter used in the query.
 * @param selectionArgs (Optional) Selection arguments used in the
 * @return The value of the _data column, which is typically a file
 */  public  static  String getDataColumn(Context context,  Uri ur
 * @param uri The Uri to check.
 * @return Whether the Uri authority is ExternalStorageProvider.
 */  public  static  boolean isExternalStorageDocument(Uri uri)  {
 * @param uri The Uri to check.
 * @return Whether the Uri authority is DownloadsProvider.
 */  public  static  boolean isDownloadsDocument(Uri uri)  {  retur
 * @param uri The Uri to check.
 * @return Whether the Uri authority is MediaProvider.
 */  public  static  boolean isMediaDocument(Uri uri)  {  return  '
```

本节小结：

> 好的，关于本节android存储访问框架SAF就到这里吧，没什么例子，后面用到再深入研究吧， 知道下就好，4.4后获取文件路径就简单多了~

4.5.1 Intent的基本使用

本节引言：

在上一节结束后意味着Android的四大组件我们都已经学习完毕了~，而本节我们要学习的是四大组件间的 枢纽——Intent(意图)，Android通信的桥梁，比如我们可以通过:

> - **startActivity**(Intent)/**startActivityForResult**(Intent)：来启动一个Activity
> - **startService**(Intent)/**bindService**(Intent)：来启动一个Service
> - **sendBroadcast**：发送广播到指定BroadcastReceiver
> - 另外别忘了我们在注册四大组件时，写得很多的**Intent-Filter**哦~

好吧，话不多说，开始本节内容！另外前面我们已经用过Intent了，就不在讲述概念性的东西了~ 老规矩，官方API：Intent

1.显式Intent与隐式Intent的区别

> - 显式**Intent**：通过组件名指定启动的目标组件,比如startActivity(new Intent(A.this,B.class)); 每次启动的组件只有一个~
> - 隐式显式**Intent**:不指定组件名,而指定Intent的Action,Data,或Category,当我们启动组件时, 会去匹配AndroidManifest.xml相关组件的Intent-filter,逐一匹配出满足属性的组件,当不止一个满足时, 会弹出一个让我们选择启动哪个的对话框~

2.Intent的七个属性：

1）ComponentName(组件名称)

ComponentName
(组件名称)

就是目标组件的名称,由组件所在应用程序配置文件中设置的包名+
组件的全限定类名组成,这是显式的Intent,激发的组件只有一个！
有setClass(),setClassName()或setComponent()方法设置
使用getComponent( )方法获取

```
ComponentName cn = new ComponentName(OneActivity.this,TwoActivity.class);
Intent it = new Intent( );
it.setComponent(cn);
```

PS:其实上面的代码直接写成:Intent it = new Intent(OneActivity.this,TwoActivity.class);就可以了,所以上面的写法没什么必要,理解下就可以了,另外说下Intent中的两个参数,前者是Context,上下文,其实就是包名,后面的是要启动的Activity,由此就确定了包名+类名 = 全限定类名,从而确定启动的是哪个Activity!
另外可以在第二个Activity调用getIntent().getComponent( )方法获得该对象,然后调用
getPackageName( )获得包名,getClassName( )获得类名

2）Action(动作)

Action(动作) →

一个普通的字符串,代表Intent要完成的一个抽象"动作",比如发信息的权限,而具体由哪个组件来完成,Intent并不负责!就是仅仅知道会有这个动作,谁来完成就交给Intent-filter进行筛选了！

要注意:在Java中的Action与Intent-filter中的格式是不一样的:比如:
①<action android:name = "android.intent.action.CALL"/>
②intent.setAction(Intent.CALL_ACTION);

3）Category(类别)

Category(类别) →

同样是普通的字符串,Category则用于为Action提供额外的附加类别信息,两者通常结合使用,一个Intent对象只能有一个Action,但是能有多个Category

同样在Java与Intetn-filter中的格式也是不一样的
①<category android:name = "android.intent.category.DEFAULT"/>
②intent.addCategorie(Intent.CATEGORY_DEFAULT);
可调用removeCategory()删除上次添加的种类,也可以用getCategories()方法获得当前对象所包含的全部种类

PS:Action是动作,Category是类别,为Action提供其他附加的信息,其实用的较多的是在intent-filter中使用,从而实现隐式的Intent,系统提供的相关的ACTION与CATEGORY可以自己查表:
在sdk的docs文档目录中的:reference-->android-->content-->Intent.html
当然如果看不懂英文话可以参考本文附录！

4）Data(数据)，Type(MIME类型)

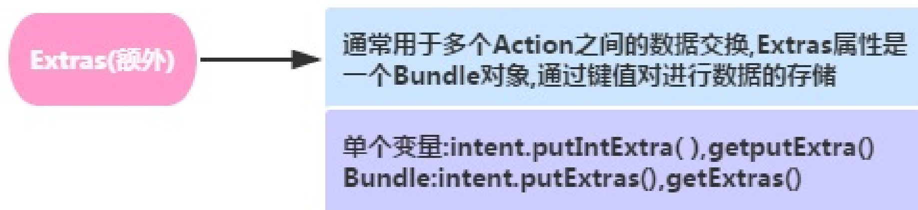
Data(数据)
Type(MIME类型) →

Data通常用于向Action属性提供操作的数据,接受一个URI对象,URI的格式:
scheme://host:port/path　　　　参数依次为:协议头,主机,端口,路径
Type通常用于指定Data所制定的Uri对应的MIME类型,比如能够显示图片数据的组建不应该应用来播放音频文件,可以是自定义的MIME类型,只要符合abc/xyz格式的字符串就可以了

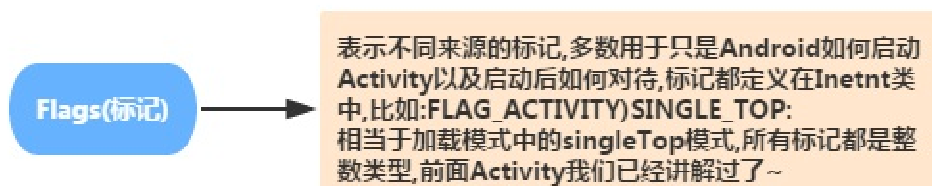
如果我们在Java代码中进行设置,另外这两个属性是会相互覆盖的
如果需要两个属性都有的话,就要调用setDataAndType()方法进行设置
而在AndroidManifest.xml文件中,这两个属性都是存放在data标签中的:

```
<data
   android:mineType = "Intent的Type属性"
   android:scheme = "Data的scheme协议头"
   android:host = "Data的主机号"
   android:port = "Data的端口号"
   android:path = "Data的路径"
   android:pathPrefix = "Data的path前缀"
   android:pathPattern = "Data属性的path的字符串模板" />
```

5）Extras(额外)



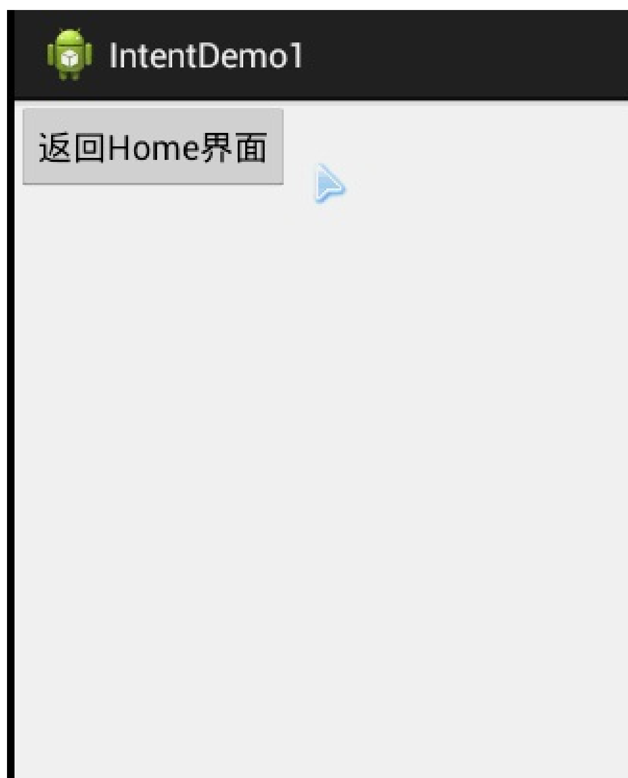
6）Flags(标记)



3.显式Intent使用示例：

这个用得很多，直接就上例子了：

例子**1**：点击按钮返回Home界面： 运行效果图：



核心代码：

```
Intent it = new Intent();
it.setAction(Intent.ACTION_MAIN);
it.addCategory(Intent.CATEGORY_HOME);
startActivity(it);
```

例子**2**：点击按钮打开百度页面： 运行效果图：



核心代码：

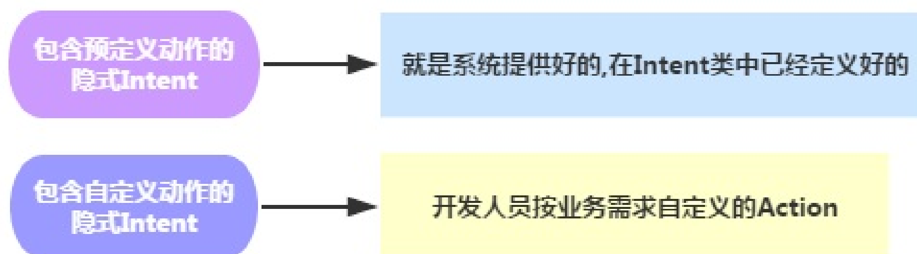
```
Intent it = new Intent();
it.setAction(Intent.ACTION_VIEW);
it.setData(Uri.parse("http://www.baidu.com"));
startActivity(it);
```

4.隐式Intent详解

隐式Intent详解

通过比较Intent对象内容与Intent-filter过滤器来实现的主要匹配的是这三个属性:
动作,数据(URI和MIME),种类
而Extras和flags在决定哪个组件时并不起任何作用

```
<intent-filter>
  <action android:name = "com.jay.example.TEST_ACTION"/>
  <category android:name = "com.intent.category.DEFAULT"/>
  <data android:mimeType = "video/mpeg" amdroid:scheme = "http".../>
</intent-filter>
```



1）预定义动作的隐式Intent示例：

代码示例:点击按钮后,所有Action为VIEW的Activity被筛选出来,由用户进一步选择:

核心代码：

建立第二个Activity的布局,与对应的Activity,在第一个Activity的按钮点击事件中添加一下代码:

```
Intent it = new Intent();
it.setAction(Intent.ACTION_VIEW);
startActivity(it);
```

最后在第二个Activity的Intent中添加以下代码:

```
<activity android:name=".SecondActivity"
            android:label="第二个Activity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
 </activity>
```

运行效果图：



2）自定义动作的隐式Intent示例：

代码示例：使用自定义的Action与category来激活另一个Activity

核心代码： 建立第二个Activity的布局,与对应的Activity,在第一个Activity的按钮点击事件中添加一下代码:

```
Intent it = new Intent();
it.setAction("my_action");
it.addCategory("my_category");
startActivity(it);
```

最后在第二个Activity的Intent中添加以下代码:

```
<activity android:name=".SecondActivity"
        android:label="第二个Activity">
    <intent-filter>
        <action android:name="my_action"/>
        <category android:name="my_category"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

注意虽然我们自定义了一个category,但是还是要把这个默认的加上,不然会报错的：

```
<category android:name="android.intent.category.DEFAULT"/>
```

5.常用系统Intent合集

大家贴下常用的系统Intent的合集吧，上面没有的欢迎提出~

```
//================================================================
//1.拨打电话
// 给移动客服10086拨打电话
Uri uri = Uri.parse("tel:10086");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);

//================================================================

//2.发送短信
// 给10086发送内容为“Hello”的短信
Uri uri = Uri.parse("smsto:10086");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
intent.putExtra("sms_body", "Hello");
startActivity(intent);

//3.发送彩信（相当于发送带附件的短信）
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra("sms_body", "Hello");
Uri uri = Uri.parse("content://media/external/images/media/23");
intent.putExtra(Intent.EXTRA_STREAM, uri);
intent.setType("image/png");
startActivity(intent);

//================================================================

//4.打开浏览器:
// 打开百度主页
Uri uri = Uri.parse("http://www.baidu.com");
Intent intent  = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//================================================================

//5.发送电子邮件:(阉割了Google服务的没戏!!!!)
// 给someone@domain.com发邮件
Uri uri = Uri.parse("mailto:someone@domain.com");
Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
startActivity(intent);
// 给someone@domain.com发邮件发送内容为“Hello”的邮件
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, "someone@domain.com");
intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
```

```
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
intent.setType("text/plain");
startActivity(intent);
// 给多人发邮件
Intent intent=new Intent(Intent.ACTION_SEND);
String[] tos = {"1@abc.com", "2@abc.com"}; // 收件人
String[] ccs = {"3@abc.com", "4@abc.com"}; // 抄送
String[] bccs = {"5@abc.com", "6@abc.com"}; // 密送
intent.putExtra(Intent.EXTRA_EMAIL, tos);
intent.putExtra(Intent.EXTRA_CC, ccs);
intent.putExtra(Intent.EXTRA_BCC, bccs);
intent.putExtra(Intent.EXTRA_SUBJECT, "Subject");
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
intent.setType("message/rfc822");
startActivity(intent);

//================================================================

//6.显示地图:
// 打开Google地图中国北京位置（北纬39.9，东经116.3）
Uri uri = Uri.parse("geo:39.9,116.3");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//================================================================

//7.路径规划
// 路径规划：从北京某地（北纬39.9，东经116.3）到上海某地（北纬31.2，东经121.
Uri uri = Uri.parse("http://maps.google.com/maps?f=d&saddr=39.9 116
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//================================================================

//8.多媒体播放:
Intent intent = new Intent(Intent.ACTION_VIEW);
Uri uri = Uri.parse("file:///sdcard/foo.mp3");
intent.setDataAndType(uri, "audio/mp3");
startActivity(intent);

//获取SD卡下所有音频文件,然后播放第一首=-=
Uri uri = Uri.withAppendedPath(MediaStore.Audio.Media.INTERNAL_CONT
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//================================================================

//9.打开摄像头拍照:
// 打开拍照程序
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, 0);
// 取出照片数据
Bundle extras = intent.getExtras();
```

```
Bitmap bitmap = (Bitmap) extras.get("data");

//另一种:
//调用系统相机应用程序，并存储拍下来的照片
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
time = Calendar.getInstance().getTimeInMillis();
intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(new File(Envi
.getExternalStorageDirectory().getAbsolutePath()+"/tucue", time + '
startActivityForResult(intent, ACTIVITY_GET_CAMERA_IMAGE);

//=================================================================

//10.获取并剪切图片
// 获取并剪切图片
Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
intent.setType("image/*");
intent.putExtra("crop", "true"); // 开启剪切
intent.putExtra("aspectX", 1); // 剪切的宽高比为1：2
intent.putExtra("aspectY", 2);
intent.putExtra("outputX", 20); // 保存图片的宽和高
intent.putExtra("outputY", 40);
intent.putExtra("output", Uri.fromFile(new File("/mnt/sdcard/temp")
intent.putExtra("outputFormat", "JPEG");// 返回格式
startActivityForResult(intent, 0);
// 剪切特定图片
Intent intent = new Intent("com.android.camera.action.CROP");
intent.setClassName("com.android.camera", "com.android.camera.CropI
intent.setData(Uri.fromFile(new File("/mnt/sdcard/temp")));
intent.putExtra("outputX", 1); // 剪切的宽高比为1：2
intent.putExtra("outputY", 2);
intent.putExtra("aspectX", 20); // 保存图片的宽和高
intent.putExtra("aspectY", 40);
intent.putExtra("scale", true);
intent.putExtra("noFaceDetection", true);
intent.putExtra("output", Uri.parse("file:///mnt/sdcard/temp"));
startActivityForResult(intent, 0);

//=================================================================

//11.打开Google Market
// 打开Google Market直接进入该程序的详细页面
Uri uri = Uri.parse("market://details?id=" + "com.demo.app");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);

//=================================================================

//12.进入手机设置界面:
// 进入无线网络设置界面（其它可以举一反三）
Intent intent = new Intent(android.provider.Settings.ACTION_WIRELES
startActivityForResult(intent, 0);

//=================================================================
```

```
//13.安装apk:
Uri installUri = Uri.fromParts("package", "xxx", null);
returnIt = new Intent(Intent.ACTION_PACKAGE_ADDED, installUri);

//=================================================================

//14.卸载apk:
Uri uri = Uri.fromParts("package", strPackageName, null);
Intent it = new Intent(Intent.ACTION_DELETE, uri);
startActivity(it);

//=================================================================

//15.发送附件:
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.putExtra(Intent.EXTRA_STREAM, "file:///sdcard/eoe.mp3");
sendIntent.setType("audio/mp3");
startActivity(Intent.createChooser(it, "Choose Email Client"));

//=================================================================

//16.进入联系人页面:
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setData(People.CONTENT_URI);
startActivity(intent);

//=================================================================

//17.查看指定联系人:
Uri personUri = ContentUris.withAppendedId(People.CONTENT_URI, info
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setData(personUri);
startActivity(intent);

//=================================================================

//18.调用系统编辑添加联系人（高版本SDK有效）：
Intent it = newIntent(Intent.ACTION_INSERT_OR_EDIT);
it.setType("vnd.android.cursor.item/contact");
//it.setType(Contacts.CONTENT_ITEM_TYPE);
it.putExtra("name","myName");
it.putExtra(android.provider.Contacts.Intents.Insert.COMPANY, "orga
it.putExtra(android.provider.Contacts.Intents.Insert.EMAIL,"email")
it.putExtra(android.provider.Contacts.Intents.Insert.PHONE,"homePho
it.putExtra(android.provider.Contacts.Intents.Insert.SECONDARY_PHON
it.putExtra( android.provider.Contacts.Intents.Insert.TERTIARY_PHON
it.putExtra(android.provider.Contacts.Intents.Insert.JOB_TITLE,"tit
startActivity(it);
```

```
//==========================================================================

//19.调用系统编辑添加联系人（全有效）：
Intent intent = newIntent(Intent.ACTION_INSERT_OR_EDIT);
intent.setType(People.CONTENT_ITEM_TYPE);
intent.putExtra(Contacts.Intents.Insert.NAME, "My Name");
intent.putExtra(Contacts.Intents.Insert.PHONE, "+1234567890");
intent.putExtra(Contacts.Intents.Insert.PHONE_TYPE,Contacts.Phones(
intent.putExtra(Contacts.Intents.Insert.EMAIL, "com@com.com");
intent.putExtra(Contacts.Intents.Insert.EMAIL_TYPE, Contacts.Contac
startActivity(intent);

//==========================================================================

//20.打开另一程序
Intent i = new Intent();
ComponentName cn = new ComponentName("com.example.jay.test",
"com.example.jay.test.MainActivity");
i.setComponent(cn);
i.setAction("android.intent.action.MAIN");
startActivityForResult(i, RESULT_OK);

//==========================================================================

//21.打开录音机
Intent mi = new Intent(Media.RECORD_SOUND_ACTION);
startActivity(mi);

//==========================================================================

//22.从google搜索内容
Intent intent = new Intent();
intent.setAction(Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY,"searchString")
startActivity(intent);

//==========================================================================
```

6.Action在哪里查？

本来想直接贴以前收集到的Intent Action的，后来想想还是算了，授之以鱼，还不如授之以渔， 如果你下载了Android的文档的话，可以在下述路径：

sdk-->docs-->reference-->android--->content--->Intent.html

找到这个玩意，然后从这个Constants开始就是了：

Constants

public static final String ACTION_AIRPLANE_MODE_CHANGED

Added in API level 1

Broadcast Action: The user has switched the phone into or out of Airpl

- *state* - A boolean value indicating whether Airplane Mode is on. I

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.AIRPLANE_MODE"

public static final String ACTION_ALL_APPS

遇到陌生的自己来这里查即可~

本节小结：

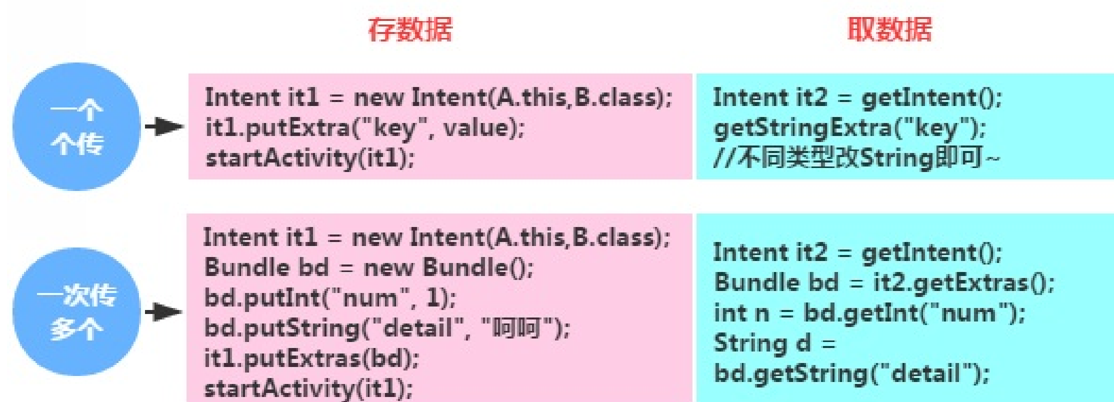
> 好的，关于Intent的基本使用就到这里，下一节我们会来继续学习在日常开发中使用Intent可能会遇到 的一些问题或者说需求吧，敬请期待，谢谢~

4.5.2 Intent之复杂数据的传递

本节引言：

> 上一节中我们学习了Intent的一些基本使用，知道了Intent的七个属性，显式Intent以及隐式Intent，以及如何自定义隐式Intent，最后还给大家提供了一些常用的系统Intent！而本节跟大家讲解的是Intent传递数据的问题~好的，开始本节内容~

1.Intent传递简单数据

> 还记得我们在Activity那里学过如何在两个Activity中互相传递简单数据的方法吗？
>
> 
>
>
> 就是可以直接通过调用Intent的putExtra()方法存入数据，然后在获得Intent后调用getXxxExtra获得对应类型的数据；传递多个的话，可以使用Bundle对象作为容器，通过调用Bundle的putXxx先将数据存储到Bundle中，然后调用Intent的putExtras()方法将Bundle存入Intent中，然后获得Intent以后，调用getExtras()获得Bundle容器，然后调用其getXXX获取对应的数据！另外数据存储有点类似于Map的<键，值>！

2.Intent传递数组

> 嘿嘿，普通类型倒没问题，但是如果是数组咧？解决方法如下：

写入数组：

```
bd.putStringArray("StringArray", new String[]{"呵呵","哈哈"});
//可把StringArray换成其他数据类型,比如int,float等等...
```

读取数组：

```
String[] str = bd.getStringArray("StringArray")
```

3.Intent传递集合

> 嗯，数组很简单吧，那我们再来传下集合~这个就稍微复杂点了，分情况处理：

1）List<基本数据类型或String>

写入集合：

```
intent.putStringArrayListExtra(name, value)
intent.putIntegerArrayListExtra(name, value)
```

读取集合：

```
intent.getStringArrayListExtra(name)
intent.getIntegerArrayListExtra(name)
```

2）List< Object>

将list强转成Serializable类型,然后传入(可用Bundle做媒介)

写入集合：

```
putExtras(key, (Serializable)list)
```

读取集合：

```
(List<Object>) getIntent().getSerializable(key)
```

PS:Object类需要实现Serializable接口

3）Map<String, Object>,或更复杂的

解决方法是：外层套个**List**

```
//传递复杂些的参数
Map<String, Object> map1 = new HashMap<String, Object>();
map1.put("key1", "value1");
map1.put("key2", "value2");
List<Map<String, Object>> list = new ArrayList<Map<String, Object>>
list.add(map1);

Intent intent = new Intent();
intent.setClass(MainActivity.this,ComplexActivity.class);
Bundle bundle = new Bundle();

//须定义一个list用于在budnle中传递需要传递的ArrayList<Object>,这个是必须要
ArrayList bundlelist = new ArrayList();
bundlelist.add(list);
bundle.putParcelableArrayList("list",bundlelist);
intent.putExtras(bundle);
startActivity(intent);
```

4.Intent传递对象

传递对象的方式有两种：将对象转换为Json字符串或者通过Serializable,Parcelable序列化 不建议使用Android内置的抠脚Json解析器，可使用fastjson或者Gson第三方库！

1）将对象转换为Json字符串

Gson解析的例子：

Model:

```
public class Author{
    private int id;
    private String name;
    //...
}

public class Author{
    private int id;
    private String name;
    //...
}
```

写入数据：

```
Book book=new Book();
book.setTitle("Java编程思想");
Author author=new Author();
author.setId(1);
author.setName("Bruce Eckel");
book.setAuthor(author);
Intent intent=new Intent(this,SecondActivity.class);
intent.putExtra("book",new Gson().toJson(book));
startActivity(intent);
```

读取数据：

```
String bookJson=getIntent().getStringExtra("book");
Book book=new Gson().fromJson(bookJson,Book.class);
Log.d(TAG,"book title->"+book.getTitle());
Log.d(TAG,"book author name->"+book.getAuthor().getName());
```

2）使用**Serializable,Parcelable**序列化对象

1.Serializable实现:

> ①业务Bean实现：Serializable接口,写上getter和setter方法 ②Intent通过调用putExtra(String name, Serializable value)传入对象实例 当然对象有多个的话多个的话,我们也可以先Bundle.putSerializable(x,x); ③新Activity调用getSerializableExtra()方法获得对象实例: eg:Product pd = (Product) getIntent().getSerializableExtra("Product"); ④调用对象get方法获得相应参数

2.Parcelable实现:

一般流程:

> ①业务Bean继承Parcelable接口,重写writeToParcel方法,将你的对象序列化为一个Parcel对象; ②重写describeContents方法，内容接口描述，默认返回0就可以 ③实例化静态内部对象CREATOR实现接口Parcelable.Creator ④同样式通过Intent的putExtra()方法传入对象实例,当然多个对象的话,我们可以先 放到Bundle里Bundle.putParcelable(x,x),再Intent.putExtras()即可

一些解释:

> 通过writeToParcel将你的对象映射成Parcel对象，再通过createFromParcel将Parcel对象映射 成你的对象。也可以将Parcel看成是一个流，通过writeToParcel把对象写到流里面， 在通过createFromParcel从流里读取对象，只不过这个过程需要你来实现，因此写的 顺序和读的顺序必须一致。

实现**Parcelable**接口的代码示例:

```
//Internal Description Interface,You do not need to manage
@Override
public int describeContents() {
     return 0;
}

@Override
public void writeToParcel(Parcel parcel, int flags){
    parcel.writeString(bookName);
    parcel.writeString(author);
    parcel.writeInt(publishTime);
}

public static final Parcelable.Creator<Book> CREATOR = new Creator<
    @Override
    public Book[] newArray(int size) {
        return new Book[size];
    }

    @Override
    public Book createFromParcel(Parcel source) {
        Book mBook = new Book();
        mBook.bookName = source.readString();
        mBook.author = source.readString();
        mBook.publishTime = source.readInt();
        return mBook;
    }
};
```

Android Studio生成Parcleable插件：

Intellij/Andriod Studio插件android-parcelable-intellij-plugin 只要ALT+Insert，即可直接生成Parcleable接口代码。

另外：Android中大量用到Parcelable对象，实现Parcable接口又是非常繁琐的,可以用到 第三方的开源框架:Parceler,因为Maven的问题,暂时还没试。

参考地址:[Android的Parcelable自动生成]

3.两种序列化方式的比较：

两者的比较:

- 1）在使用内存的时候，Parcelable比Serializable性能高，所以推荐使用Parcelable。
- 2）Serializable在序列化的时候会产生大量的临时变量，从而引起频繁的GC。
- 3）Parcelable不能使用在要将数据存储在磁盘上的情况，因为Parcelable不能很好的保证数据的 持续性在外界有变化的情况下。尽管Serializable效率低点，但此时还是建议使用Serializable。

5.Intent传递Bitmap

bitmap默认实现Parcelable接口,直接传递即可

实现代码：

```
Bitmap bitmap = null;
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putParcelable("bitmap", bitmap);
intent.putExtra("bundle", bundle);
```

6.传来传去不方便，直接定义全局数据

> 如果是传递简单的数据，有这样的需求，Activity1 -> Activity2 -> Activity3 -> Activity4， 你想在Activity中传递某个数据到Activity4中，怎么破，一个个页面传么？
>
> 显然不科学是吧，如果你想某个数据可以在任何地方都能获取到，你就可以考虑使用 **Application**全局对象了！
>
> Android系统在每个程序运行的时候创建一个Application对象，而且只会创建一个，所以Application 是单例(singleton)模式的一个类，而且Application对象的生命周期是整个程序中最长的，他的生命 周期等于这个程序的生命周期。如果想存储一些比静态的值(固定不改变的，也可以变)，如果你想使用 Application 就需要自定义类实现Application类，并且告诉系统实例化的是我们自定义的 Application 而非系统默认的，而这一步，就是在AndroidManifest.xml中卫我们的application标签添加:**name**属性！

关键部分代码：

1）自定义**Application**类：

```
class MyApp extends Application {
    private String myState;
    public String getState(){
        return myState;
    }
    public void setState(String s){
        myState = s;
    }
}
```

2）**AndroidManifest.xml**中声明：

```
<application android:name=".MyApp" android:icon="@drawable/icon"
  android:label="@string/app_name">
```

3）在需要的地方调用：

```
class Blah extends Activity {
    @Override
    public void onCreate(Bundle b){
        ...
    MyApp appState = ((MyApp)getApplicationContext());
    String state = appState.getState();
        ...
    }
}
```

高逼格写法

：在任何位置都能获取到Application全局对象。

Applicaiton是系统的一个组件，他也有自己的一个生命周期，我们可以在onCraete里获得这个 Application对象。贴下修改后的代码吧！

```
class MyApp extends Application {
    private String myState;
    private static MyApp instance;

    public static MyApp getInstance(){
        return instance;
    }

    public String getState(){
        return myState;
    }
    public void setState(String s){
        myState = s;
    }

    @Override
    public void onCreate(){
        onCreate();
        instance = this;
    }

}
```

然后在任意地方我们就可以直接调用：MyApp.getInstance（）来获得Application的全局对象！

注意事项：

> Application对象是存在于内存中的，也就有它可能会被系统杀死，比如这样的场景：
>
> 我们在Activity1中往application中存储了用户账号，然后在Activity2中获取到用户账号，并且显示！
>
> 如果我们点击home键，然后过了N久候，系统为了回收内存kill掉了我们的app。这个时候，我们重新 打开这个app，这个时候很神奇的，回到了Activity2的页面，但是如果这个时候你再去获取Application 里的用户账号，程序就会报NullPointerException，然后crash掉~
>
> 之所以会发生上述crash，是因为这个Application对象是全新创建的，可能你以为App是重新启动的， 其实并不是，仅仅是创建一个新的Application，然后启动上次用户离开时的Activity，从而创造App 并没有被杀死的假象！所以如果是比较重要的数据的话，建议你还是进行本地化，另外在使用数据的时候 要对变量的值进行非空检查！还有一点就是：不止是Application变量会这样，单例对象以及公共静态变量 也会这样~

7.单例模式传参

> 上面的Application就是基于单例的，单例模式的特点就是可以保证系统中一个类有且只有一个实例。 这样很容易就能实现，在A中设置参数，在B中直接访问了。这是几种方法中效率最高的。

范例代码：(代码来自于网上~)

①定义一个单例类：

```
public class XclSingleton
{
    //单例模式实例
    private static XclSingleton instance = null;

    //synchronized 用于线程安全，防止多线程同时创建实例
    public synchronized static XclSingleton getInstance(){
        if(instance == null){
            instance = new XclSingleton();
        }
        return instance;
    }

    final HashMap<String, Object> mMap;
    private XclSingleton()
    {
        mMap = new HashMap<String,Object>();
    }

    public void put(String key,Object value){
        mMap.put(key,value);
    }

    public Object get(String key)
    {
        return mMap.get(key);
    }

}
```

②设置参数:

```
XclSingleton.getInstance().put("key1", "value1");
XclSingleton.getInstance().put("key2", "value2");
```

本节小结：

> 好的，关于Intent复杂数据传输就到这里，本节除了讲述使用Intent来传递复杂数据外，还教了大家 使用Application和单例模式来传递参数！相信会对大家在数据传递方面带来方便，谢谢~

5.1 Fragment基本概述

本节引言

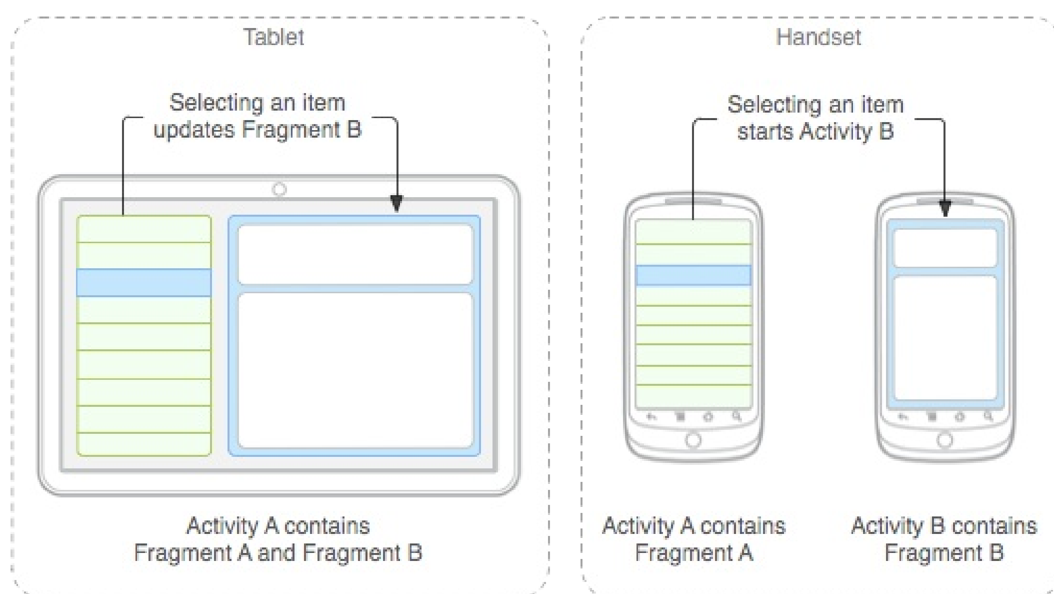
> 好的，在上一章中我们把Android的四大组件Activity，Service，BroadCastReceiver，ContentProvider 以及他们之间的纽带：Intent，都撸了一遍，而本章节给大家带来的是一个Fragment(碎片)的东西， 本节我们就来介绍这个Fragment的一些基本概念以及用法！ 官方文档：Fragment

1.基本概念

1）它是什么鬼，有什么用？

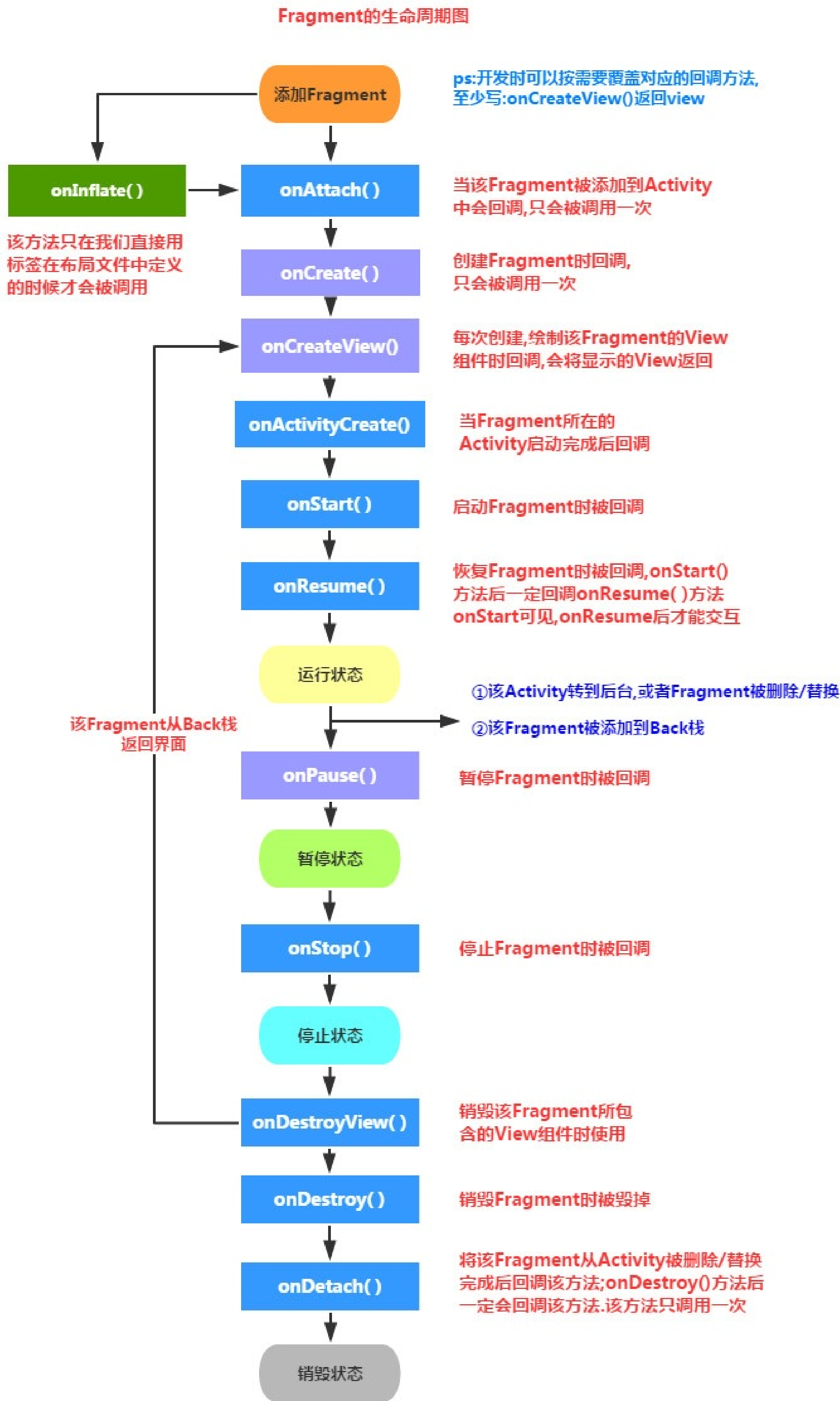
> 答：Fragment是Android3.0后引入的一个新的API，他出现的初衷是为了适应大屏幕的平板电脑， 当然现在他仍然是平板APP UI设计的宠儿，而且我们普通手机开发也会加入这个Fragment， 我们可以把他看成一个小型的Activity，又称Activity片段！想想，如果一个很大的界面，我们 就一个布局，写起界面来会有多麻烦，而且如果组件多的话是管理起来也很麻烦！而使用Fragment 我们可以把屏幕划分成几块，然后进行分组，进行一个模块化的管理！从而可以更加方便的在 运行过程中动态地更新Activity的用户界面！另外Fragment并不能单独使用，他需要嵌套在Activity 中使用，尽管他拥有自己的生命周期，但是还是会受到宿主Activity的生命周期的影响，比如Activity 被destory销毁了，他也会跟着销毁！

下图是文档中给出的一个Fragment分别对应手机与平板间不同情况的处理图：



PS:简单的新闻浏览页面，使用两个Fragment分别显示新闻列表与新闻内容；

2）Fragment的生命周期图


Fragment的生命周期图
添加Fragment
ps:开发时可以按需要覆盖对应的回调方法,
至少写:onCreateView()返回view
onInflate()
该方法只在我们直接用
标签在布局文件中定义
的时候才会被调用
onAttach()
当该Fragment被添加到Activity
中会回调,只会被调用一次
onCreate()
创建Fragment时回调,
只会被调用一次
onCreateView()
每次创建,绘制该Fragment的View
组件时回调,会将显示的View返回
onActivityCreate()
当Fragment所在的
Activity启动完成后回调
onStart()
启动Fragment时被回调
onResume()
恢复Fragment时被回调,onStart()
方法后一定回调onResume()方法
onStart可见,onResume后才能交互
运行状态
①该Activity转到后台,或者Fragment被删除/替换
②该Fragment被添加到Back栈
该Fragment从Back栈
返回界面
onPause()
暂停Fragment时被回调
暂停状态
onStop()
停止Fragment时被回调
停止状态
onDestroyView()
销毁该Fragment所包
含的View组件时使用
onDestroy()
销毁Fragment时被毁掉
onDetach()
将该Fragment从Activity被删除/替换
完成后回调该方法;onDestroy()方法后
一定会回调该方法.该方法只调用一次
销毁状态

3）核心要点：

下面说下使用Fragment的一些要点：

> - 3.0版本后引入,即minSdk要大于11
> - Fragment需要嵌套在Activity中使用,当然也可以嵌套到另外一个Fragment中,但这个被嵌套 的Fragment也是需要嵌套在Activity中的,间接地说,Fragment还是需要嵌套在Activity中!! 受寄主Activity的生命周期影响,当然他也有自己的生命周期! 另外不建议在Fragment里面 嵌套Fragment因为嵌套在里面的Fragment生命周期不可控!!!
> - 官方文档说创建Fragment时至少需要实现三个方法:onCreate( ),onCreateView( ),OnPause( ); 不过貌似只写一个onCreateView也是可以的...
> - Fragment的生命周期和Activity有点类似:三种状态: Resumed:在允许中的Fragment可见 Paused:所在Activity可见,但是得不到焦点 Stoped: ①调用addToBackStack(),Fragment被添加到Bcak栈 ②该Activity转向后台,或者该Fragment被替换/删除 ps:停止状态的fragment仍然活着(所有状态和成员信息被系统保持着),然而,它对用户 不再可见,并且如果activity被干掉,他也会被干掉.

4）Fragment的几个子类：

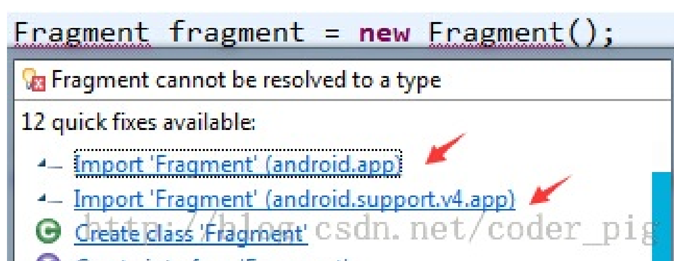
ps:很多时候我们都是直接重写Fragment,inflate加载布局完成相应业务了,子类用的不多,等需要的 时候在深入研究!

> - 对话框:**DialogFragment**
> - 列表:**ListFragment**
> - 选项设置:**PreferenceFragment**
> - WebView界面:**WebViewFragment**

5）是用App包下的Fragment还是v4包下的：

问题概述：

相信很多朋友在使用Fragment的时候都会遇到下面这种情况：



那么我们到底是使用android.app下的Fragment还是用的android.support.v4.app包下 的Fragment呢？

> 答：其实都可以，前面说过Fragment是Android 3.0(API 11)后引入的，那么如果开发的app需要 在3.0以下的版本运行呢?比如还有一点点市场份额的2.3!于是乎,v4包就这样应运而生了, 而最低可以兼容到1.6版本！至于使用哪个包看你的需求了,现在3.0下手机市场份额其实已经不多了,随街都是4.0以上的，6.0十月份都出了，你说呢...所以这个时候,你可以直接使用app包下的Fragment 然后调用相关的方法，通常都是不会有什么问题的;如果你Fragment用了app包的, FragmentManager和FragmentTransaction都需要是app包的！要么用全部用app,要么全部用v4, 不然可是会报错的哦!当然如果你要自己的app对于低版本的手机也兼容的话,那么就可以选择用v4包！

使用**v4**包下**Fragment**要注意的地方：

> - ①如果你使用了v4包下的Fragment,那么所在的那个Activity就要继承FragmentActivity哦! 案例:今天在xml文件中静态地载入fragment,然后重写了Fragment,但是在加载Activity的时候就报错了， 大概的提示就是Fragment错误还是找不到什么的,name属性改了几次还是错!最后才发现是用了 v4的包的缘故,只需让自己的Activity改成FragmentActivity即可!
> - ②之前写了下面这段代码，然后报错：
>
> ```
> getFragmentManager().beginTransaction().replace(R.id.fragment1, fragment);
> ```
>
> The method replace(int, Fragment) in the type FragmentTransaction is not applicable for the arguments (int, Fragment)
> 1 quick fix available:
> Change type of 'fragment' to 'Fragment'
> http://blog.csdn.net/coder_pig
>
> 有点莫名其妙啊,Fragment,FragmentManager,FragmentTransaction都是用的v4包啊, Activity也是继承FragmentActivity的啊?都改成app包就可以了,但是这不和我们用v4包的 前提冲突了么?其实也是有解决方法的哈? 答:只需要把**getFragmentManager**( )改成**getSupportFragmentManager**( )就可以了

2.创建一个Fragment

1）静态加载Fragment

实现流程：



示例代码：

Step 1:定义Fragment的布局，就是fragment显示内容的 **Step 2:**自定义一个Fragment类,需要继承Fragment或者他的子类,重写onCreateView()方法 在该方法中调用:inflater.inflate()方法加载Fragment的布局文件,接着返回加载的view对象

```
public class Fragmentone extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment1, container,
        return view;
    }
}
```

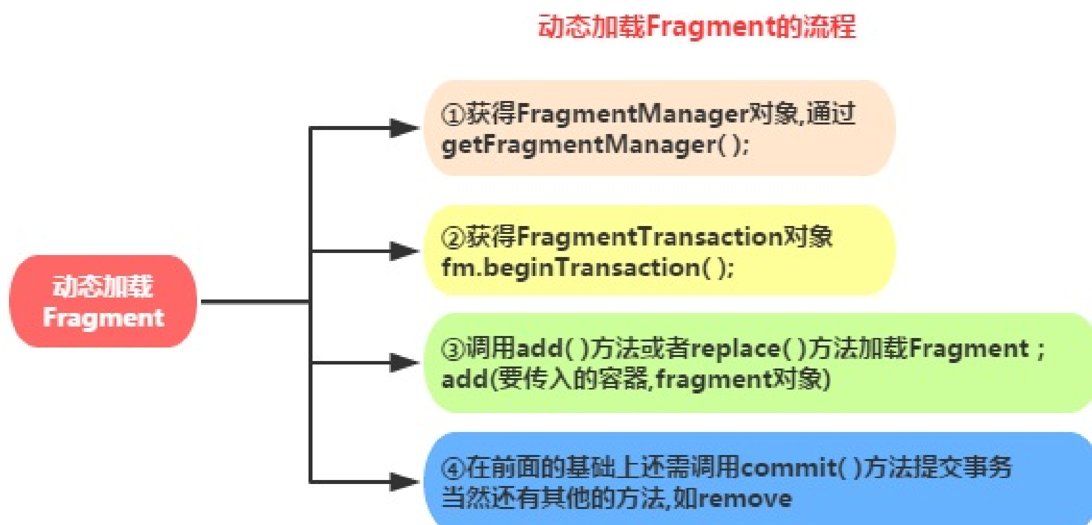
Step 3:在需要加载Fragment的Activity对应的布局文件中添加fragment的标签， 记住，name属性是全限定类名哦，就是要包含Fragment的包名，如:

```
<fragment
    android:id="@+id/fragment1"
    android:name="com.jay.example.fragmentdemo.Fragmentone"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />
```

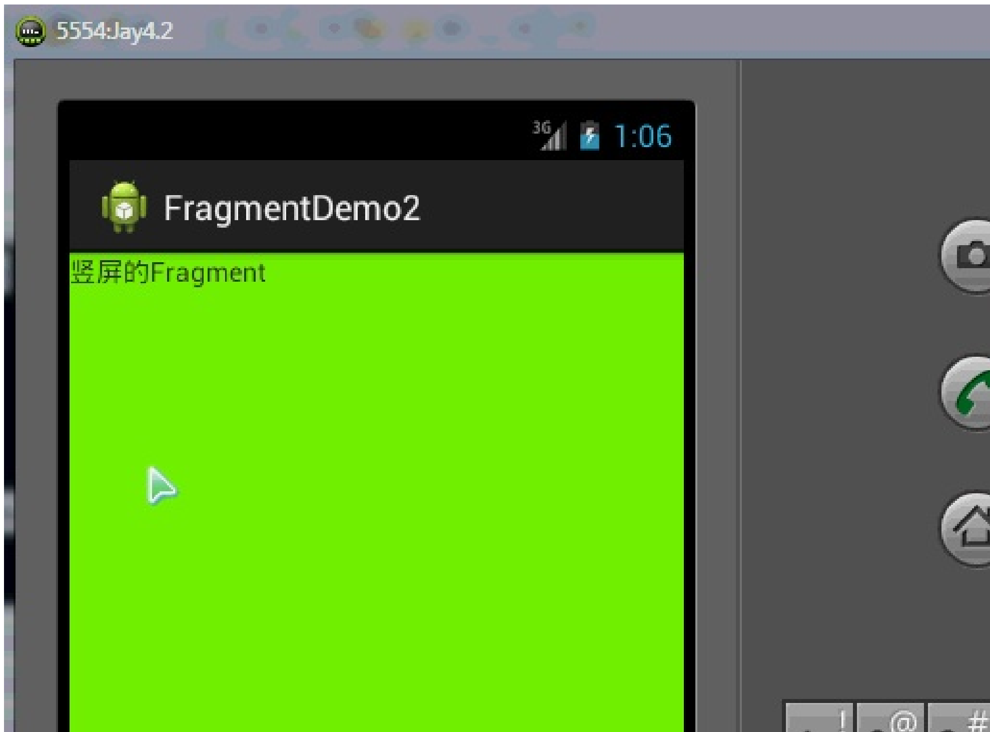
Step 4: Activity在onCreate( )方法中调用setContentView()加载布局文件即可!

2）动态加载Fragment

实现流程：



示例代码： 这里演示的是，当横竖屏切换的时候地切换Fragment：



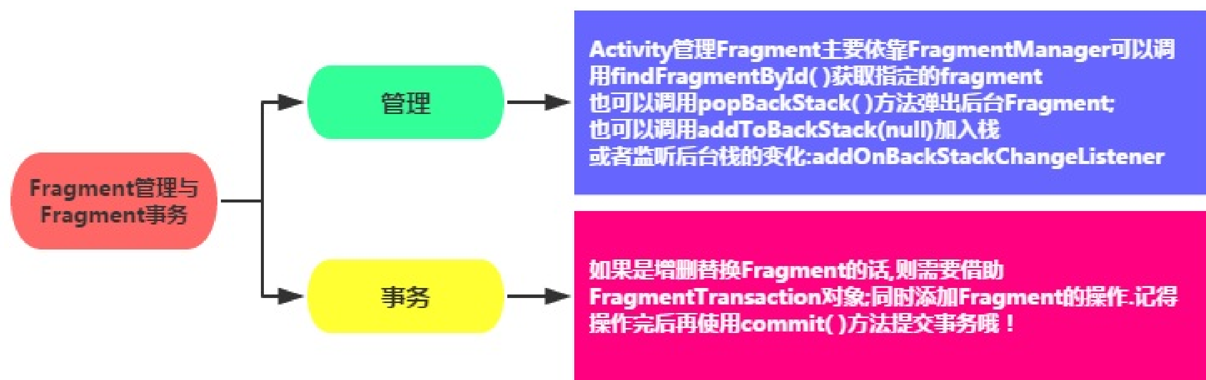
Fragment以及布局代码就不贴出来了，直接贴MainActivity的关键代码：

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Display dis = getWindowManager().getDefaultDisplay();
        if(dis.getWidth() > dis.getHeight())
        {
            Fragment1 f1 = new Fragment1();
            getFragmentManager().beginTransaction().replace(R.id.L
        }

        else
        {
            Fragment2 f2 = new Fragment2();
            getFragmentManager().beginTransaction().replace(R.id.L
        }
    }
}
```

3.Fragment管理与Fragment事务



4.Fragment与Activity的交互



可能有的朋友不喜欢看图，接下来用文字介绍下吧：

1）组件获取

Fragment获得Activity中的组件: **getActivity().findViewById(R.id.list)**； Activity获得Fragment中的组件(根据id和tag都可以)：**getFragmentManager.findFragmentByid(R.id.fragment1);**

2）数据传递

①Activit传递数据给Fragment:

> 在Activity中创建Bundle数据包,调用Fragment实例的setArguments(bundle) 从而将Bundle数据包传给Fragment,然后Fragment中调用getArguments获得Bundle对象,然后进行解析就可以了

②Fragment传递数据给Activity

> 在Fragment中定义一个内部回调接口,再让包含该Fragment的Activity实现该回调接口, Fragment就可以通过回调接口传数据了,回调,相信很多人都知道是什么玩意,但是 写不出来啊,网上的一百度"fragment传数据给Activity",全是李刚老师的那个代码,真心无语 算了,这里就写下局部代码吧,相信读者一看就懂的了:

Step 1:定义一个回调接口:**(Fragment**中**)**

```
/*接口*/
public interface CallBack{
    /*定义一个获取信息的方法*/
    public void getResult(String result);
}
```

Step 2：接口回调（**Fragment**中）

```
/*接口回调*/
public void getData(CallBack callBack){
    /*获取文本框的信息,当然你也可以传其他类型的参数,看需求咯*/
    String msg = editText.getText().toString();
    callBack.getResult(msg);
}
```

Step 3:使用接口回调方法读数据**(Activity**中**)**

```
/* 使用接口回调的方法获取数据 */
leftFragment.getData(new CallBack() {
 @Override
       public void getResult(String result) {              /*打印信
            Toast.makeText(MainActivity.this, "-->>" + result, 1).s
            }
        });
```

总结下方法： **->**在Fragment定义一个接口,接口中定义抽象方法,你要传什么类型的数据参数就设置为什么类型; **->**接着还有写一个调用接口中的抽象方法,把要传递的数据传过去 **->**再接着就是Activity了,调用Fragment提供的那个方法,然后重写抽象方法的时候进行数据 的读取就可以了!!!

③Fragment与**Fragment**之间的数据互传

> 其实这很简单,找到要接受数据的fragment对象,直接调用setArguments传数据进去就可以了 通常的话是replace时,即fragment跳转的时候传数据的,那么只需要在初始化要跳转的Fragment 后调用他的setArguments方法传入数据即可! 如果是两个Fragment需要即时传数据,而非跳转的话,就需要先在Activity获得f1传过来的数据, 再传到f2了,就是以Activity为媒介~

示例代码如下：

```
FragmentManager fManager = getSupportFragmentManager( );
FragmentTransaction fTransaction = fManager.beginTransaction();
Fragmentthree t1 = new Fragmentthree();
Fragmenttwo t2 = new Fragmenttwo();
Bundle bundle = new Bundle();
bundle.putString("key",id);
t2.setArguments(bundle);
fTransaction.add(R.id.fragmentRoot, t2, "~~~");
fTransaction.addToBackStack(t1);
fTransaction.commit();
```

5.走一次生命周期图：

思前想后还是决定要带大家简单的走一趟生命周期图，加深大家对Fragment生命周期的理解：

> ①Activity加载Fragment的时候,依次调用下面的方法: **onAttach** -> **onCreate** -> **onCreateView** -> **onActivityCreated** -> **onStart** ->**onResume**
>
> ②当我们弄出一个悬浮的对话框风格的Activity,或者其他,就是让Fragment所在的Activity可见,但不获得焦点 **onPause**
>
> ③当对话框关闭,Activity又获得了焦点: **onResume**
>
> ④当我们替换Fragment,并调用addToBackStack()将他添加到Back栈中 **onPause -> onStop -> onDestoryView** ！！注意,此时的Fragment还没有被销毁哦!!!
>
> ⑤当我们按下键盘的回退键，Fragment会再次显示出来: **onCreateView -> onActivityCreated -> onStart -> onResume**
>
> ⑥如果我们替换后,在事务commit之前没有调用**addToBackStack()**方法将Fragment添加到back栈中的话;又或者退出了Activity的话,那么Fragment将会被完全结束, Fragment会进入销毁状态 **onPause** -> **onStop** -> **onDestoryView** -> **onDestory** -> **onDetach**

本节小结：

> 本节跟大家讲解了以下Fragment一些基本的概念以及简单的用法，相信大家会慢慢喜欢上 Fragment的，因为篇幅的关系，本节就写这么多，下一节我们带大家来写一些关于Fragment 的常用实例，敬请期待，谢谢~

5.2.1 Fragment实例精讲——底部导航栏的实现(方法1)

本节引言：

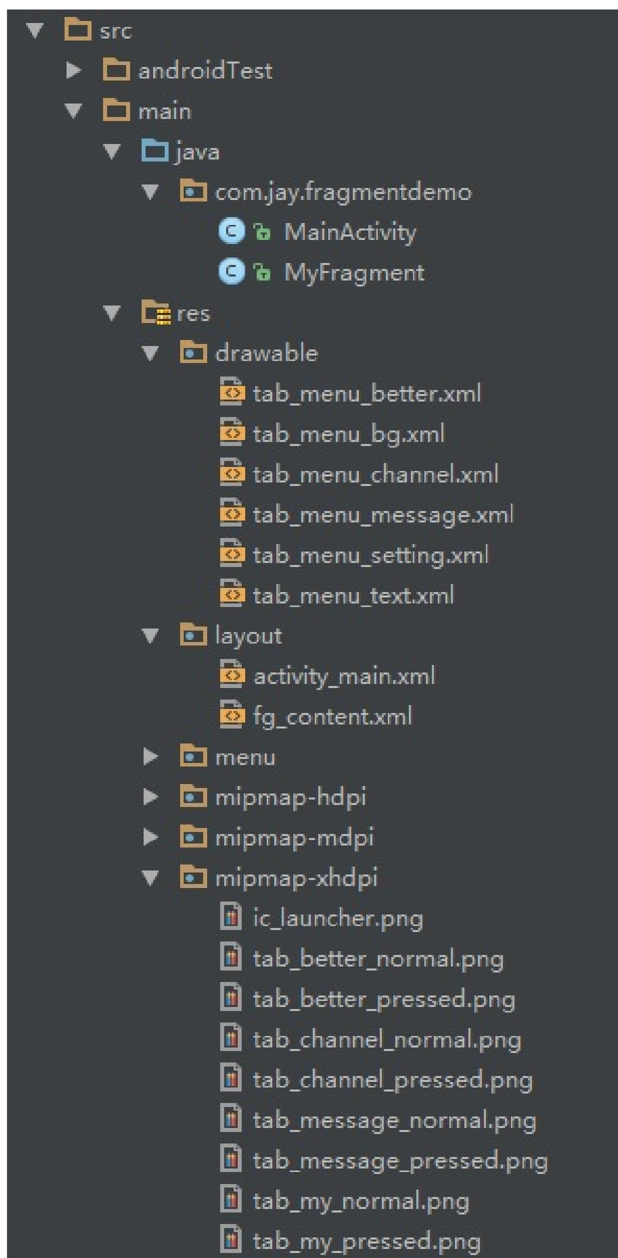
> 在上一节中我们对Fragment进行了一个初步的了解，学习了概念，生命周期，Fragment管理与 Fragment事务，以及动态与静态加载Fragment。从本节开始我们会讲解一些Fragment在实际开发 中的一些实例！而本节给大家讲解的是底部导航栏的实现！而基本的底部导航栏方法有很多种， 比如全用TextView做，或者用RadioButton，又或者使用TabLayout + RadioButton，当然复杂 的情况还是得走外层套布局的方法！本节我们用TextView来做一个底部导航栏的效果，也熟悉 下Fragment的使用！好的，开始本节内容！

1.要实现的效果图以及工程目录结构：

先看看效果图吧：



接着看看我们的工程的目录结构：



2.实现流程：

Step 1：写下底部选项的一些资源文件

> 我们从图上可以看到，我们底部的每一项点击的时候都有不同的效果是吧！ 我们是通过是否selected来判定的！我们要写的资源文件有：首先是图片，然后是文字，接着是背景！

图片Drawable资源：**tab_menu_channel.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androi
    <item android:drawable="@mipmap/tab_channel_pressed" android:st
    <item android:drawable="@mipmap/tab_channel_normal" />
</selector>
```

其他三个照葫芦画瓢！

文字资源：**tab_menu_text.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androi
    <item android:color="@color/text_yellow" android:state_selecte
    <item android:color="@color/text_gray" />
</selector>
```

背景资源：**tab_menu_bg.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androi
    <item android:state_selected="true">
        <shape>
            <solid android:color="#FFC4C4C4" />
        </shape>
    </item>
    <item>
        <shape>
            <solid android:color="@color/transparent" />
        </shape>
    </item>
</selector>
```

Step 2：编写我们的Activity布局

activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <RelativeLayout
        android:id="@+id/ly_top_bar"
```

```
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="@color/bg_topbar">

        <TextView
            android:id="@+id/txt_topbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_centerInParent="true"
            android:gravity="center"
            android:textSize="18sp"
            android:textColor="@color/text_topbar"
            android:text="信息"/>

        <View
            android:layout_width="match_parent"
            android:layout_height="2px"
            android:background="@color/div_white"
            android:layout_alignParentBottom="true"/>

    </RelativeLayout>

    <LinearLayout
        android:id="@+id/ly_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_alignParentBottom="true"
        android:background="@color/bg_white"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/txt_channel"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/tab_menu_bg"
            android:drawablePadding="3dp"
            android:drawableTop="@drawable/tab_menu_channel"
            android:gravity="center"
            android:padding="5dp"
            android:text="@string/tab_menu_alert"
            android:textColor="@drawable/tab_menu_text"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/txt_message"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/tab_menu_bg"
            android:drawablePadding="3dp"
            android:drawableTop="@drawable/tab_menu_message"
            android:gravity="center"
```

```
            android:padding="5dp"
            android:text="@string/tab_menu_profile"
            android:textColor="@drawable/tab_menu_text"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/txt_better"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/tab_menu_bg"
            android:drawablePadding="3dp"
            android:drawableTop="@drawable/tab_menu_better"
            android:gravity="center"
            android:padding="5dp"
            android:text="@string/tab_menu_pay"
            android:textColor="@drawable/tab_menu_text"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/txt_setting"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/tab_menu_bg"
            android:drawablePadding="3dp"
            android:drawableTop="@drawable/tab_menu_setting"
            android:gravity="center"
            android:padding="5dp"
            android:text="@string/tab_menu_setting"
            android:textColor="@drawable/tab_menu_text"
            android:textSize="16sp"/>

    </LinearLayout>

    <View
        android:id="@+id/div_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="2px"
        android:background="@color/div_white"
        android:layout_above="@id/ly_tab_bar"/>

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/ly_top_bar"
        android:layout_above="@id/div_tab_bar"
        android:id="@+id/ly_content">

    </FrameLayout>

</RelativeLayout>
```

代码解析：

> 首先定义顶部标题栏的样式，48dp的LinearLayout中间加上一个TextView作为标题！ 接着定义一个大小为56dp的LinerLayout对其底部，在这个里面加入四个TextView，比例1:1:1:1， 并且设置相关属性，接着在这个LinearLayout上加一条线段！ 最后以标题栏和底部导航栏为边界，写一个FrameLayout，宽高match_parent，用做Fragment的容器！

PS：这里四个TextView属性是重复的，你也可以自行抽取出来，编写一个style，设置下~

Step 3：隐藏顶部导航栏

意外发现以前的在Activity中调用**requestWindowFeature(Window.FEATURE_NO_TITLE)**;可以隐藏手机 自带顶部导航栏，但是写demo时候发现会报错，即使这句话写在了setContentView()之前！可能是因为 继承的是AppCompatActivity而非Activity类！ 当然以前的getSupportActionbar().hide()隐藏掉Actionbar，但是他还是会在界面上！ 最后还有一种方法就是自己编写一个style，然后在AndroidManifest.xml中为Application设置这个Theme：

> 注： 把 requestWindowFeature(Window.FEATURE_NO_TITLE);放在super.onCreate(savedInstanceState);前面就可以隐藏ActionBar而不报错。

接着**AndroidManifest.xml**设置下theme属性：

```
android:theme="@style/Theme.AppCompat.NoActionBar"
```

PS：上述"良心代码"由好程序员曹神赞助~

Step 4：创建一个Fragment的简单布局与类：

fg_content.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:orientation="vertical" android:layout_width="match_pare
    android:layout_height="match_parent"
    android:background="@color/bg_white">

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="呵呵"
        android:textColor="@color/text_yellow"
        android:textSize="20sp"/>

</LinearLayout>
```

MyFragment.java:

```
/**
 * Created by Coder-pig on 2015/8/28 0028.
 */
public class MyFragment extends Fragment {

    private String content;
    public MyFragment(String content) {
        this.content = content;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_content,container,
        TextView txt_content = (TextView) view.findViewById(R.id.t
        txt_content.setText(content);
        return view;
    }
}
```

代码解析：

就是简单的重写了一个onCreateView()方法，其他方法可以按需重写!

Step 5：编写MainActivity.java

先说说我们要考虑的一些关键问题：

> - Fragment什么时候初始化和add到容器中？什么时候hide和show？
> - 如何让TextView被选中？选中一个TextView后，要做一些什么操作？
> - 刚进入MainActivity怎么样让一个TextView处于Selected的状态？

嗯，接下来一一解答上面这些问题：

> - 我们选中TextView后对对应的Fragment进行判空，如果为空，初始化，并添加到容器中；而hide的话，我们定义一个方法hide所有的Fragment，每次触发点击事件就先调用这个hideAll方法，讲所有Fragment隐藏起来，然后如果TextView对应的Fragment不为空，我们就将这个Fragment显示出来；
> - 这个我们通过点击事件来实现，点击TextView后先重置所有TextView的选中状态为false，然后设置点击的 TextView的选中状态为true；
> - 这个更简单，我们是通过点击事件来设置选中的，那么在onCreate()方法里加个触发点击事件的 方法不就可以了嘛~ **txt_channel.performClick();**

逻辑都弄懂了，直接上代码咯：

MainActivity.java:

```
/**
 * Created by Coder-pig on 2015/8/28 0028.
 */
public class MainActivity extends AppCompatActivity implements View

    //UI Object
    private TextView txt_topbar;
    private TextView txt_channel;
    private TextView txt_message;
    private TextView txt_better;
    private TextView txt_setting;
    private FrameLayout ly_content;

    //Fragment Object
    private MyFragment fg1,fg2,fg3,fg4;
    private FragmentManager fManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);
        fManager = getFragmentManager();
        bindViews();
        txt_channel.performClick();   //模拟一次点击，既进去后选择第一项
    }

    //UI组件初始化与事件绑定
    private void bindViews() {
        txt_topbar = (TextView) findViewById(R.id.txt_topbar);
        txt_channel = (TextView) findViewById(R.id.txt_channel);
        txt_message = (TextView) findViewById(R.id.txt_message);
```

```
        txt_better = (TextView) findViewById(R.id.txt_better);
        txt_setting = (TextView) findViewById(R.id.txt_setting);
        ly_content = (FrameLayout) findViewById(R.id.ly_content);

        txt_channel.setOnClickListener(this);
        txt_message.setOnClickListener(this);
        txt_better.setOnClickListener(this);
        txt_setting.setOnClickListener(this);
    }

    //重置所有文本的选中状态
    private void setSelected(){
        txt_channel.setSelected(false);
        txt_message.setSelected(false);
        txt_better.setSelected(false);
        txt_setting.setSelected(false);
    }

    //隐藏所有Fragment
    private void hideAllFragment(FragmentTransaction fragmentTransa
        if(fg1 != null)fragmentTransaction.hide(fg1);
        if(fg2 != null)fragmentTransaction.hide(fg2);
        if(fg3 != null)fragmentTransaction.hide(fg3);
        if(fg4 != null)fragmentTransaction.hide(fg4);
    }

    @Override
    public void onClick(View v) {
        FragmentTransaction fTransaction = fManager.beginTransactio
        hideAllFragment(fTransaction);
        switch (v.getId()){
            case R.id.txt_channel:
                setSelected();
                txt_channel.setSelected(true);
                if(fg1 == null){
                    fg1 = new MyFragment("第一个Fragment");
                    fTransaction.add(R.id.ly_content,fg1);
                }else{
                    fTransaction.show(fg1);
                }
                break;
            case R.id.txt_message:
                setSelected();
                txt_message.setSelected(true);
                if(fg2 == null){
                    fg2 = new MyFragment("第二个Fragment");
                    fTransaction.add(R.id.ly_content,fg2);
                }else{
                    fTransaction.show(fg2);
                }
                break;
            case R.id.txt_better:
                setSelected();
```

```
                txt_better.setSelected(true);
                if(fg3 == null){
                    fg3 = new MyFragment("第三个Fragment");
                    fTransaction.add(R.id.ly_content,fg3);
                }else{
                    fTransaction.show(fg3);
                }
                break;
            case R.id.txt_setting:
                setSelected();
                txt_setting.setSelected(true);
                if(fg4 == null){
                    fg4 = new MyFragment("第四个Fragment");
                    fTransaction.add(R.id.ly_content,fg4);
                }else{
                    fTransaction.show(fg4);
                }
                break;
        }
        fTransaction.commit();
    }
}
```

3.代码下载：

FragmentDemo.zip：FragmentDemo.zip 下载 声明：图片素材来自 App：**better**，本代码只做演示，并无用于商业用途！

4.本节小结

> 本节给大家讲解了如何使用一个LinarLayout + 四个TextView 实现一个底部导航栏以及 Fragment add，hide，show的逻辑~还是蛮简单的，最后要感谢小猪秘密基地的基神，B神， 还有好程序员曹神给我的一些指点！万分感谢，仅以此篇纪念小猪重返装逼界，嗯，重 返应用层，嘿嘿，本节就到这里，谢谢~

5.2.2 Fragment实例精讲——底部导航栏的实现(方法2)

本节引言：

> 上一节中我们使用LinearLayout + TextView实现了底部导航栏的效果，每次点击我们都要重置 所有TextView的状态，然后选中点击的TextView，有点麻烦是吧，接下来我们用另一种方法： RadioGroup + RadioButton来实现我们上一节的效果！

1.一些碎碎念

本节用到的是实现单选效果的RadioButton，如果你不熟悉，或者没用过，可先移步到：RadioButton 简单点说就是我们就是一个RadioGroup包着四个RadioButton，和前面一样用比例来划分:1:1:1:1；另外我们只需重写RadioGroup的onCheckedChange，判断checkid即可知道点击的是哪个RadioButton！ 好的，下面开始堆码！

2.实现流程

PS:这里的素材什么的，直接使用的是上一节中的素材！另外drawable类的资源都是将selected 状态修改成checked！

Step 1：写底部选项的一些资源文件

图片Drawable资源：**tab_menu_channel.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:drawable="@mipmap/tab_channel_pressed" android:st
    <item android:drawable="@mipmap/tab_channel_normal" />
</selector>
```

其他三个照葫芦画瓢！

文字资源：**tab_menu_text.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:color="@color/text_yellow" android:state_checked=
    <item android:color="@color/text_gray" />
</selector>
```

背景资源：**tab_menu_bg.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_selected="true">
        <shape>
            <solid android:color="#FFC4C4C4" />
        </shape>
    </item>
    <item>
        <shape>
            <solid android:color="@color/transparent" />
        </shape>
    </item>
</selector>
```

Step 2：编写我们的Activity布局

> 在前面用TextView实现底部导航栏我们就发现了一个问题，每个TextView的属性都几乎是差不多 的，而在建议那里我们也说让大家把相同的属性抽取出来写到Style中，可能部分朋友懒或者不知道 如何抽取出来，以及用，这里就给大家示范下：

首先我们取出其中一个RadioGroup的标签：

```
<RadioButton
            android:id="@+id/rb_channel"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:background="@drawable/tab_menu_bg"
            android:button="@null"
            android:drawableTop="@drawable/tab_menu_channel"
            android:gravity="center"
            android:paddingTop="3dp"
            android:text="@string/tab_menu_alert"
            android:textColor="@drawable/tab_menu_text"
            android:textSize="18sp" />
```

我们可以把每个RadioButton都相同的属性抽取出来，写到**style.xml**文件中：

```
<style name="tab_menu_item">
    <item name="android:layout_width">0dp</item>
    <item name="android:layout_weight">1</item>
    <item name="android:layout_height">match_parent</item>
    <item name="android:background">@drawable/tab_menu_bg</item>
    <item name="android:button">@null</item>
    <item name="android:gravity">center</item>
    <item name="android:paddingTop">3dp</item>
    <item name="android:textColor">@drawable/tab_menu_text</item>
    <item name="android:textSize">18sp</item>
</style>
```

然后我们的activity_main.xml中的RadioButton就用不着次次都写相同的代码了，只需让RadioButton的**style="@style/tab_menu_item"**就可以了！

activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/bg_gray"
    tools:context=".MainActivity">

    <RelativeLayout
        android:id="@+id/ly_top_bar"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="@color/bg_topbar">

        <TextView
            android:id="@+id/txt_topbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_centerInParent="true"
            android:gravity="center"
            android:text="信息"
            android:textColor="@color/text_topbar"
            android:textSize="18sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="2px"
            android:layout_alignParentBottom="true"
            android:background="@color/div_white" />

    </RelativeLayout>
```

```
    <RadioGroup
        android:id="@+id/rg_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_alignParentBottom="true"
        android:background="@color/bg_white"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/rb_channel"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_channel"
            android:text="@string/tab_menu_alert" />

        <RadioButton
            android:id="@+id/rb_message"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_message"
            android:text="@string/tab_menu_profile" />

        <RadioButton
            android:id="@+id/rb_better"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_better"
            android:text="@string/tab_menu_pay" />

        <RadioButton
            android:id="@+id/rb_setting"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_setting"
            android:text="@string/tab_menu_setting"/>

    </RadioGroup>

    <View
        android:id="@+id/div_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="2px"
        android:layout_above="@id/rg_tab_bar"
        android:background="@color/div_white" />

    <FrameLayout
        android:id="@+id/ly_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/div_tab_bar"
        android:layout_below="@id/ly_top_bar"></FrameLayout>

</RelativeLayout>
```

Step 3：隐藏顶部导航栏

AndroidManifest.xml设置下**theme**属性

```
android:theme="@style/Theme.AppCompat.NoActionBar"
```

Step 4：创建一个Fragment的简单布局与类：

直接照搬上一节的布局跟Fragment：

fg_content.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:orientation="vertical" android:layout_width="match_pare
    android:layout_height="match_parent"
    android:background="@color/bg_white">

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="呵呵"
        android:textColor="@color/text_yellow"
        android:textSize="20sp"/>

</LinearLayout>
```

MyFragment.java:

```
/**
 * Created by Coder-pig on 2015/8/29 0028.
 */
public class MyFragment extends Fragment {

    private String content;
    public MyFragment(String content) {
        this.content = content;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_content,container,
        TextView txt_content = (TextView) view.findViewById(R.id.tx
        txt_content.setText(content);
        return view;
    }
}
```

Step 5：编写MainActivity.java

这个比起TextView实现简单多了，就不详细讲解了，很简单，直接上代码：

MainActivity.java

```
/**
 * Created by Coder-pig on 2015/8/29 0028.
 */
public class MainActivity extends AppCompatActivity implements Radi

    private RadioGroup rg_tab_bar;
    private RadioButton rb_channel;

    //Fragment Object
    private MyFragment fg1,fg2,fg3,fg4;
    private FragmentManager fManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        fManager = getFragmentManager();
        rg_tab_bar = (RadioGroup) findViewById(R.id.rg_tab_bar);
        rg_tab_bar.setOnCheckedChangeListener(this);
        //获取第一个单选按钮，并设置其为选中状态
        rb_channel = (RadioButton) findViewById(R.id.rb_channel);
        rb_channel.setChecked(true);
    }
```

```
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        FragmentTransaction fTransaction = fManager.beginTransactio
        hideAllFragment(fTransaction);
        switch (checkedId){
            case R.id.rb_channel:
                if(fg1 == null){
                    fg1 = new MyFragment("第一个Fragment");
                    fTransaction.add(R.id.ly_content,fg1);
                }else{
                    fTransaction.show(fg1);
                }
                break;
            case R.id.rb_message:
                if(fg2 == null){
                    fg2 = new MyFragment("第二个Fragment");
                    fTransaction.add(R.id.ly_content,fg2);
                }else{
                    fTransaction.show(fg2);
                }
                break;
            case R.id.rb_better:
                if(fg3 == null){
                    fg3 = new MyFragment("第三个Fragment");
                    fTransaction.add(R.id.ly_content,fg3);
                }else{
                    fTransaction.show(fg3);
                }
                break;
            case R.id.rb_setting:
                if(fg4 == null){
                    fg4 = new MyFragment("第四个Fragment");
                    fTransaction.add(R.id.ly_content,fg4);
                }else{
                    fTransaction.show(fg4);
                }
                break;
        }
        fTransaction.commit();
    }

    //隐藏所有Fragment
    private void hideAllFragment(FragmentTransaction fragmentTransa
        if(fg1 != null)fragmentTransaction.hide(fg1);
        if(fg2 != null)fragmentTransaction.hide(fg2);
        if(fg3 != null)fragmentTransaction.hide(fg3);
        if(fg4 != null)fragmentTransaction.hide(fg4);
    }

}
```

PS:在上一节忘记讲一点了，FragmentTransaction只能使用一次，每次使用都要调用FragmentManager 的beginTransaction()方法获得FragmentTransaction事务对象哦！

3.运行效果图

其实和上一节实现的效果是一样的：



4.代码下载：

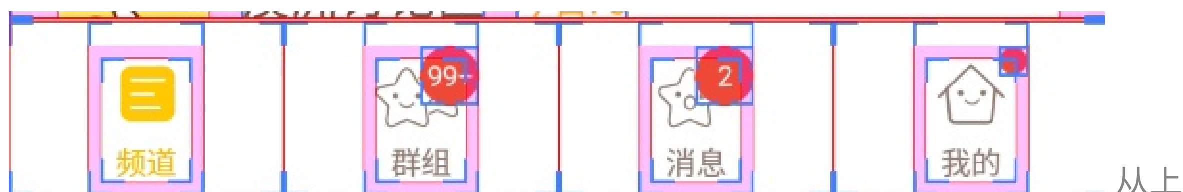
FragmentDemo2.zip：FragmentDemo2.zip 下载

5.本节小结：

> 本节讲解的是实现底部导航栏的第二种方法:RadioGroup + RadioButton，有了单选，我们就不用像TextView一样，每次点击后先重置所有TextView的Selected状态，再让点击的TextView 的Selected为true，这样就可以写少一点代码了~本节就到这里~谢谢

5.2.3 Fragment实例精讲——底部导航栏的实现(方法3)

本节引言

> 前面我们已经跟大家讲解了实现底部导航栏的两种方案，但是这两种方案只适合普通的情况，如果 是像新浪微博那样的，想在底部导航栏上的item带有一个红色的小点，然后加上一个消息数目这样， 前面两种方案就显得无力了，我们来看看别人的APP是怎么做的，打开手机的开发者选项，勾选里面的： 显示布局边界，然后打开我们参考的那个App，可以看到底部导航栏是这样的：
>
> 
>
>
> 从上面这个图我们就可以看出，这种底部导航栏不是简单的TextView或者RadioGroup构成的， 大概布局方案可能是：外层一个LinearLayout，中间一个RelativeLayout，而在中间有一个TextView， 然后再在TextView的右上角有一个红色圆圈背景的TextView或者一个红色的小点； 大概就这样，而这些小点平时的时候应该设置的不可见，当收到信息推送，即有相关类别信息的 时候再可见，并且显示对应的信息数目！那么下面我们就来实现下这种底部导航栏的效果， 另外，为了方便演示，这里就不演示Fragment的切换效果了！另外顺道复习下Fragment获得Activity 中的组件的知识点！

1.实现效果图：

为了方便理解，这里通过点击按钮的形式，模拟收到推送信息，然后显示红色点！

运行效果图：



2.实现流程：

好的，接下来我们就来实现上面这个效果~

Step 1：相关资源文件的准备：

和前面一样，准备好drawable系列的资源：

文字资源：**tab_menu_text.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:color="@color/text_yellow" android:state_selectec
    <item android:color="@color/text_gray" />
</selector>
```

图标资源：**tab_menu_better.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:drawable="@mipmap/tab_better_pressed" android:sta
    <item android:drawable="@mipmap/tab_better_normal" />
</selector>
```

照着把其他三个也撸出来~！

Step 2：编写activity的布局代码：

因为四个选项的TextView以及右上角的红点数字属性都差不多，如下：

```
<TextView
                    android:id="@+id/tab_menu_channel"
                    android:layout_marginTop="5dp"
                    android:layout_width="wrap_content"
                    android:layout_height="48dp"
                    android:layout_centerInParent="true"
                    android:textColor="@drawable/tab_menu_text"
                    android:drawableTop="@drawable/tab_menu_channel
                    android:text="@string/tab_menu_alert"/>
                <TextView
                    android:layout_width="20dp"
                    android:layout_height="20dp"
                    android:background="@mipmap/bg_num"
                    android:layout_toRightOf="@+id/tab_menu_channel
                    android:layout_marginLeft="-10dp"
                    android:text="99+"
                    android:textSize="12sp"
                    android:gravity="center"
                    android:textColor="@color/text_white"/>
```

我们将他们抽取出来，写到style.xml里：

```
<style name="tab_menu_text">
    <item name="android:layout_marginTop">5dp</item>
    <item name="android:layout_width">wrap_content</item>
    <item name="android:layout_height">48dp</item>
    <item name="android:layout_centerInParent">true</item>
    <item name="android:textColor">@drawable/tab_menu_text</item>
</style>

<style name="tab_menu_bgnum">
    <item name="android:layout_width">20dp</item>
    <item name="android:layout_height">20dp</item>
    <item name="android:background">@mipmap/bg_num</item>
    <item name="android:layout_marginLeft">-10dp</item>
    <item name="android:textSize">12sp</item>
    <item name="android:gravity">center</item>
    <item name="android:textColor">@color/text_white</item>
</style>
```

然后开始编写我们的activity.xml布局：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <RelativeLayout
        android:id="@+id/ly_top_bar"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="@color/bg_topbar">

        <TextView
            android:id="@+id/txt_topbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_centerInParent="true"
            android:gravity="center"
            android:text="信息"
            android:textColor="@color/text_topbar"
            android:textSize="18sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="2px"
            android:layout_alignParentBottom="true"
            android:background="@color/div_white" />

    </RelativeLayout>

    <LinearLayout
        android:id="@+id/ly_tab_menu"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_alignParentBottom="true"
        android:background="@color/bg_white"
        android:orientation="horizontal">

        <LinearLayout
            android:id="@+id/ly_tab_menu_channel"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:padding="5dp">

                <TextView
                    android:id="@+id/tab_menu_channel"
```

```
                    style="@style/tab_menu_text"
                    android:drawableTop="@drawable/tab_menu_channel
                    android:text="@string/tab_menu_alert" />

                <TextView
                    android:id="@+id/tab_menu_channel_num"
                    style="@style/tab_menu_bgnum"
                    android:layout_toRightOf="@+id/tab_menu_channel
                    android:text="99+"
                    android:visibility="gone" />
            </RelativeLayout>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/ly_tab_menu_message"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:padding="5dp">

                <TextView
                    android:id="@+id/tab_menu_message"
                    style="@style/tab_menu_text"
                    android:drawableTop="@drawable/tab_menu_message
                    android:text="@string/tab_menu_profile" />

                <TextView
                    android:id="@+id/tab_menu_message_num"
                    style="@style/tab_menu_bgnum"
                    android:layout_toRightOf="@+id/tab_menu_message
                    android:text="99+"
                    android:visibility="gone" />
            </RelativeLayout>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/ly_tab_menu_better"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:padding="5dp">

                <TextView
```

```
                    android:id="@+id/tab_menu_better"
                    style="@style/tab_menu_text"
                    android:drawableTop="@drawable/tab_menu_better'
                    android:text="@string/tab_menu_pay" />

                <TextView
                    android:id="@+id/tab_menu_better_num"
                    style="@style/tab_menu_bgnum"
                    android:layout_toRightOf="@+id/tab_menu_better'
                    android:text="99+"
                    android:visibility="gone" />
            </RelativeLayout>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/ly_tab_menu_setting"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center">

            <RelativeLayout
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:padding="5dp">

                <TextView
                    android:id="@+id/tab_menu_setting"
                    style="@style/tab_menu_text"
                    android:drawableTop="@drawable/tab_menu_setting
                    android:text="@string/tab_menu_alert" />

                <ImageView
                    android:id="@+id/tab_menu_setting_partner"
                    android:layout_width="12dp"
                    android:layout_height="12dp"
                    android:layout_marginLeft="-5dp"
                    android:layout_toRightOf="@id/tab_menu_setting'
                    android:visibility="gone"
                    android:src="@mipmap/partner_red" />

            </RelativeLayout>
        </LinearLayout>

    </LinearLayout>

    <View
        android:id="@+id/div_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="2px"
        android:layout_above="@id/ly_tab_menu"
        android:background="@color/div_white" />
```

```
    <FrameLayout
        android:id="@+id/ly_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/div_tab_bar"
        android:layout_below="@id/ly_top_bar"/>

</RelativeLayout>
```

Step 3：编写Fragment界面布局以及类

Fragment布局由四个普通按钮构成：

fg_my.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <Button
        android:id="@+id/btn_one"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="第一个显示信息"/>

    <Button
        android:id="@+id/btn_two"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="第二个显示信息"/>

    <Button
        android:id="@+id/btn_three"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="第三个显示信息"/>

    <Button
        android:id="@+id/btn_four"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="第四个显示信息"/>

</LinearLayout>
```

接着是自定义的Fragment类，这里的话我们通过getActivity.findViewById()来获得Activity 中的小红点，这里仅仅是简单的控制显示而已！ **MyFragment.java**:

```
public class MyFragment extends Fragment implements View.OnClickLis

    private Context mContext;
    private Button btn_one;
    private Button btn_two;
    private Button btn_three;
    private Button btn_four;

    public MyFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_my,container,false
        //UI Object
        btn_one = (Button) view.findViewById(R.id.btn_one);
        btn_two = (Button) view.findViewById(R.id.btn_two);
        btn_three = (Button) view.findViewById(R.id.btn_three);
        btn_four = (Button) view.findViewById(R.id.btn_four);
        //Bind Event
        btn_one.setOnClickListener(this);
        btn_two.setOnClickListener(this);
        btn_three.setOnClickListener(this);
        btn_four.setOnClickListener(this);
        return view;
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_one:
                TextView tab_menu_channel_num = (TextView) getActiv
                tab_menu_channel_num.setText("11");
                tab_menu_channel_num.setVisibility(View.VISIBLE);
                break;
            case R.id.btn_two:
                TextView tab_menu_message_num = (TextView) getActiv
                tab_menu_message_num.setText("20");
                tab_menu_message_num.setVisibility(View.VISIBLE);
                break;
            case R.id.btn_three:
                TextView tab_menu_better_num = (TextView) getActivi
                tab_menu_better_num.setText("99+");
                tab_menu_better_num.setVisibility(View.VISIBLE);
                break;
            case R.id.btn_four:
                ImageView tab_menu_setting_partner = (ImageView) ge
                tab_menu_setting_partner.setVisibility(View.VISIBLE
```

```
                    break;
            }
        }
    }
```

Step 4：编写MainActivity

我们在这里完成主要的逻辑实现，有些部分和前面TextView实现底部导航栏的效果类似， 就不具体讲解了，代码如下：

MainActivity.java：

```
/**
 * Created by Coder-pig on 2015/8/30 0030.
 */
public class MainActivity extends AppCompatActivity implements View

    //Activity UI Object
    private LinearLayout ly_tab_menu_channel;
    private TextView tab_menu_channel;
    private TextView tab_menu_channel_num;
    private LinearLayout ly_tab_menu_message;
    private TextView tab_menu_message;
    private TextView tab_menu_message_num;
    private LinearLayout ly_tab_menu_better;
    private TextView tab_menu_better;
    private TextView tab_menu_better_num;
    private LinearLayout ly_tab_menu_setting;
    private TextView tab_menu_setting;
    private ImageView tab_menu_setting_partner;
    private FragmentManager fManager;
    private FragmentTransaction fTransaction;
    private MyFragment fg1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
        ly_tab_menu_channel.performClick();
        fg1 = new MyFragment();
        fManager = getFragmentManager();
        fTransaction = fManager.beginTransaction();
        fTransaction.add(R.id.ly_content, fg1).commit();

    }

    private void bindViews() {
        ly_tab_menu_channel = (LinearLayout) findViewById(R.id.ly_t
        tab_menu_channel = (TextView) findViewById(R.id.tab_menu_ch
```

```
        tab_menu_channel_num = (TextView) findViewById(R.id.tab_mer
        ly_tab_menu_message = (LinearLayout) findViewById(R.id.ly_t
        tab_menu_message = (TextView) findViewById(R.id.tab_menu_me
        tab_menu_message_num = (TextView) findViewById(R.id.tab_mer
        ly_tab_menu_better = (LinearLayout) findViewById(R.id.ly_ta
        tab_menu_better = (TextView) findViewById(R.id.tab_menu_bet
        tab_menu_better_num = (TextView) findViewById(R.id.tab_menu
        ly_tab_menu_setting = (LinearLayout) findViewById(R.id.ly_t
        tab_menu_setting = (TextView) findViewById(R.id.tab_menu_se
        tab_menu_setting_partner = (ImageView) findViewById(R.id.ta

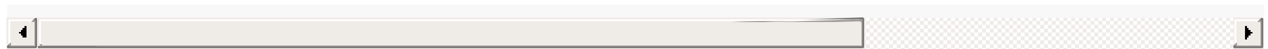
        ly_tab_menu_channel.setOnClickListener(this);
        ly_tab_menu_message.setOnClickListener(this);
        ly_tab_menu_better.setOnClickListener(this);
        ly_tab_menu_setting.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.ly_tab_menu_channel:
                setSelected();
                tab_menu_channel.setSelected(true);
                tab_menu_channel_num.setVisibility(View.INVISIBLE);
                break;
            case R.id.ly_tab_menu_message:
                setSelected();
                tab_menu_message.setSelected(true);
                tab_menu_message_num.setVisibility(View.INVISIBLE);
                break;
            case R.id.ly_tab_menu_better:
                setSelected();
                tab_menu_better.setSelected(true);
                tab_menu_better_num.setVisibility(View.INVISIBLE);
                break;
            case R.id.ly_tab_menu_setting:
                setSelected();
                tab_menu_setting.setSelected(true);
                tab_menu_setting_partner.setVisibility(View.INVISIE
                break;
        }
    }

    //重置所有文本的选中状态
    private void setSelected() {
        tab_menu_channel.setSelected(false);
        tab_menu_message.setSelected(false);
        tab_menu_better.setSelected(false);
        tab_menu_setting.setSelected(false);
    }

}
```



好的，至此，就大功告成了~

3.代码下载：

FragmentDemo3.zip：FragmentDemo3.zip下载

4.本节小结：

> 好的，本节相比前面两节稍微复杂了一点，不过还是比较容易弄懂的！另外，关于实现普通底部导航栏的实现例子就写这么多吧，下一节开始我们来写下 在此基础上的根据手势操作切换页面的例子，嗯，就说这么多，谢谢~

5.2.4 Fragment实例精讲——底部导航栏+ViewPager滑动切换页面

> 前三节我们分别用不同的方式实现了普通底部导航栏的效果，而本节我们将会在第二个实例的基础上 加上ViewPager来实现滑动切换页面的效果！大部分朋友都知道这个ViewPager是什么东西吧，如果 不知道没关系，下面我们简单的来介绍一个这个控件！

1.ViewPager简单介绍

1）是怎么样的一个控件？

> 答：一个页面切换的组件，我们可以往里面填充多个View，然后我们可以通过触摸屏幕左右滑动 切换不同的View，和前面学习的ListView一样，我们需要一个Adapter(适配器)，将要显示的View和 我们的ViewPager进行绑定，而ViewPager有他自己特定的Adapter——PagerAdapter！另外，Google 官方是建议我们使用Fragment来填充ViewPager的,这样可以更加方便的生成每个Page以及管理 每个Page的生命周期!当然它给我们提供了两个不同的Adapter，他们分别是：**FragmentPageAdapter**和**FragmentStatePagerAdapter**！ 而我们本节用到的则是前者：FragmentPageAdapter！ 另外要说一点的是**ViewPager**的缓存机制：ViewPager会缓存当前页，前一页，以及后一页，比如有1，2，3，4这四个页面： 当我们处于第一页：缓存1，2 ——> 处于第二页：缓存 1，2，3 ——> 处于第三页：缓存2，3，4 ——> 处于第四页缓存3，4这样！

2）使用PagerAdapter要重写相关方法：

> - **getCount( )**:获得viewpager中有多少个view
> - **destroyItem( )**:移除一个给定位置的页面。适配器有责任从容器中删除这个视图。这是为了确保 在finishUpdate(viewGroup)返回时视图能够被移除。
> - **instantiateItem( )**:①将给定位置的view添加到ViewGroup(容器)中,创建并显示出来 ②返回一个代表新增页面的Object(key),通常都是直接返回view本身就可以了,当然你也可以自定义自己的key,但是key和每个view要一一对应的关系
> - **isViewFromObject**( ):判断instantiateItem(ViewGroup, int)函数所返回来的Key与一个页面视图是否是 代表的同一个视图(即它俩是否是对应的，对应的表示同一个View),通常我们直接写 return view == object；就可以了,至于为什么要这样讲起来比较复杂,后面有机会进行了解吧 貌似是ViewPager中有个存储view状态信息的ArrayList,根据View取出对应信息的吧!

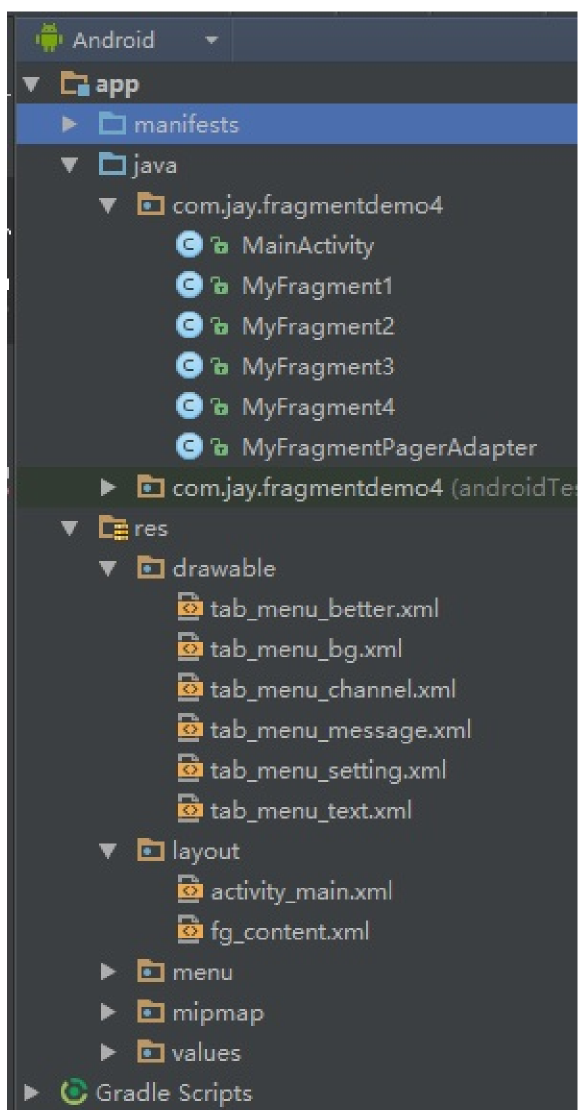
PS:不一定要重写所有方法~

2.实现效果图以及工程目录结构：

先来看下我们要实现的效果吧



接下来看下我们的项目结构：



3.代码实现：

Step 1：相关资源文件的准备：

PS：我们是在实现底部导航栏方法2的基础上来写的，所以资源文件照搬即可！ 这里就不贴多次了~！

Step 2：编写activity_main.xml的布局文件：

PS：只是把前面的FrameLayout替换成了：android.support.v4.view.ViewPager而已：

activity_mian.xml：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    tools:context=".MainActivity">

    <RelativeLayout
        android:id="@+id/ly_top_bar"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:background="@color/bg_topbar">

        <TextView
            android:id="@+id/txt_topbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_centerInParent="true"
            android:gravity="center"
            android:text="提醒"
            android:textColor="@color/text_topbar"
            android:textSize="18sp" />

        <View
            android:layout_width="match_parent"
            android:layout_height="2px"
            android:layout_alignParentBottom="true"
            android:background="@color/div_white" />

    </RelativeLayout>

    <RadioGroup
        android:id="@+id/rg_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_alignParentBottom="true"
        android:background="@color/bg_white"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/rb_channel"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_channel"
            android:text="@string/tab_menu_alert" />

        <RadioButton
            android:id="@+id/rb_message"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_message"
            android:text="@string/tab_menu_profile" />

        <RadioButton
            android:id="@+id/rb_better"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_better"
            android:text="@string/tab_menu_pay" />

        <RadioButton
```

```
            android:id="@+id/rb_setting"
            style="@style/tab_menu_item"
            android:drawableTop="@drawable/tab_menu_setting"
            android:text="@string/tab_menu_setting" />

    </RadioGroup>

    <View
        android:id="@+id/div_tab_bar"
        android:layout_width="match_parent"
        android:layout_height="2px"
        android:layout_above="@id/rg_tab_bar"
        android:background="@color/div_white" />

    <android.support.v4.view.ViewPager
        android:id="@+id/vpager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/div_tab_bar"
        android:layout_below="@id/ly_top_bar" />

</RelativeLayout>
```

Step 3：编写Fragment的布局以及代码：

PS：这里为了顺便演示ViewPager的机制，特意写成了四个Fragment！在onCreateView中打印创建Log！

fg_content.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/bg_white"
    android:orientation="vertical">

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="呵呵"
        android:textColor="@color/text_yellow"
        android:textSize="20sp" />

</LinearLayout>
```

MyFragment1.java:

```
/**
 * Created by Jay on 2015/8/28 0028.
 */
public class MyFragment1 extends Fragment {

    public MyFragment1() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_content, containe
        TextView txt_content = (TextView) view.findViewById(R.id.tx
        txt_content.setText("第一个Fragment");
        Log.e("HEHE", "1日狗");
        return view;
    }
}
```

其他三个照葫芦画瓢，换点东西就好！

Step 4：自定义FragmentPagerAdapter类 ：

代码很简单，只需传递一个FragmentManager过来，其他都在这里完成！

```
/**
 * Created by Jay on 2015/8/31 0031.
 */
public class MyFragmentPagerAdapter extends FragmentPagerAdapter {

    private final int PAGER_COUNT = 4;
    private MyFragment1 myFragment1 = null;
    private MyFragment2 myFragment2 = null;
    private MyFragment3 myFragment3 = null;
    private MyFragment4 myFragment4 = null;

    public MyFragmentPagerAdapter(FragmentManager fm) {
        super(fm);
        myFragment1 = new MyFragment1();
        myFragment2 = new MyFragment2();
        myFragment3 = new MyFragment3();
        myFragment4 = new MyFragment4();
    }

    @Override
    public int getCount() {
        return PAGER_COUNT;
    }
```

```
    @Override
    public Object instantiateItem(ViewGroup vg, int position) {
        return super.instantiateItem(vg, position);
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Obje
        System.out.println("position Destory" + position);
        super.destroyItem(container, position, object);
    }

    @Override
    public Fragment getItem(int position) {
        Fragment fragment = null;
        switch (position) {
            case MainActivity.PAGE_ONE:
                fragment = myFragment1;
                break;
            case MainActivity.PAGE_TWO:
                fragment = myFragment2;
                break;
            case MainActivity.PAGE_THREE:
                fragment = myFragment3;
                break;
            case MainActivity.PAGE_FOUR:
                fragment = myFragment4;
                break;
        }
        return fragment;
    }

}
```

Step 5：MainActivity的编写：

逻辑很简单，自己看~

MainActivity.java:

```
package com.jay.fragmentdemo4;

import android.os.Bundle;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

/**
```

```
 * Created by Coder-pig on 2015/8/28 0028.
 */
public class MainActivity extends AppCompatActivity implements Radi
        ViewPager.OnPageChangeListener {

    //UI Objects
    private TextView txt_topbar;
    private RadioGroup rg_tab_bar;
    private RadioButton rb_channel;
    private RadioButton rb_message;
    private RadioButton rb_better;
    private RadioButton rb_setting;
    private ViewPager vpager;

    private MyFragmentPagerAdapter mAdapter;

    //几个代表页面的常量
    public static final int PAGE_ONE = 0;
    public static final int PAGE_TWO = 1;
    public static final int PAGE_THREE = 2;
    public static final int PAGE_FOUR = 3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mAdapter = new MyFragmentPagerAdapter(getSupportFragmentMan
        bindViews();
        rb_channel.setChecked(true);
    }

    private void bindViews() {
        txt_topbar = (TextView) findViewById(R.id.txt_topbar);
        rg_tab_bar = (RadioGroup) findViewById(R.id.rg_tab_bar);
        rb_channel = (RadioButton) findViewById(R.id.rb_channel);
        rb_message = (RadioButton) findViewById(R.id.rb_message);
        rb_better = (RadioButton) findViewById(R.id.rb_better);
        rb_setting = (RadioButton) findViewById(R.id.rb_setting);
        rg_tab_bar.setOnCheckedChangeListener(this);

        vpager = (ViewPager) findViewById(R.id.vpager);
        vpager.setAdapter(mAdapter);
        vpager.setCurrentItem(0);
        vpager.addOnPageChangeListener(this);
    }

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        switch (checkedId) {
            case R.id.rb_channel:
                vpager.setCurrentItem(PAGE_ONE);
                break;
            case R.id.rb_message:
```

```
                vpager.setCurrentItem(PAGE_TWO);
                break;
            case R.id.rb_better:
                vpager.setCurrentItem(PAGE_THREE);
                break;
            case R.id.rb_setting:
                vpager.setCurrentItem(PAGE_FOUR);
                break;
        }
    }

    //重写ViewPager页面切换的处理方法
    @Override
    public void onPageScrolled(int position, float positionOffset,
    }

    @Override
    public void onPageSelected(int position) {
    }

    @Override
    public void onPageScrollStateChanged(int state) {
        //state的状态有三个，0表示什么都没做，1正在滑动，2滑动完毕
        if (state == 2) {
            switch (vpager.getCurrentItem()) {
                case PAGE_ONE:
                    rb_channel.setChecked(true);
                    break;
                case PAGE_TWO:
                    rb_message.setChecked(true);
                    break;
                case PAGE_THREE:
                    rb_better.setChecked(true);
                    break;
                case PAGE_FOUR:
                    rb_setting.setChecked(true);
                    break;
            }
        }
    }
}
```

PS：嘿嘿，上面我把导包部分的代码也贴出来了，就是害怕你们导错包，然后出现一些莫名其妙的错误！ 因为ViewPager是v4包下面的，所以Fragment，FragmentManager，FragmentTransaction都是需要使用 V4包下的哦！另外获取FragmentManager的方法不是直接用getFragmentManager()而是使用getSupportFragmentManager()哦！

4.代码下载：

FragmentDemo4：[下载FragmentDemo4.zip]

本节小结：

> 好的，上面就是底部导航栏 + ViewPager实现简单滑动切换Fragment的实现过程了！ 就说这么多，谢谢~

5.2.5 Fragment实例精讲——新闻(购物)类App列表Fragment的简单实现

本节引言：

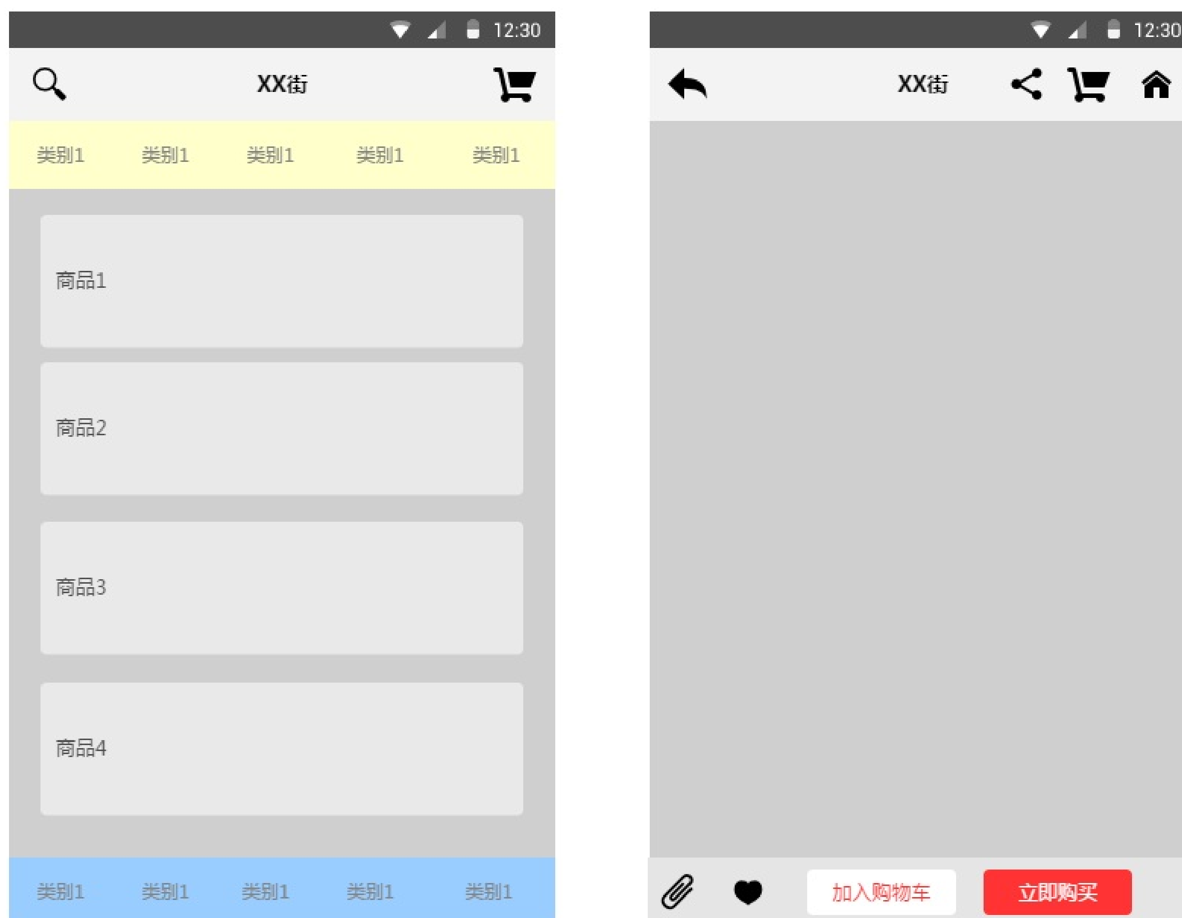
> 相信大家对点击列表，然后进入详情这种App并不陌生吧，在购物类App和新闻类App中最为常见：下面我们简单来讲一下流程逻辑！

1.逻辑流程讲解：

刚好公司测试妹子的测试机上装了楚楚街9块9的APP，呵呵，直接就照这个来研究吧：



嘿嘿，市面上很多APP都是这种样子的，而这个可以用我们学到的Fragment来实现：可能gif动画看不清，笔者用界面原型工具画个大概吧：



大概就这样，中间区域是一个布局容器，一般是FrameLayout，然后我们将一个Fragment replace 到这个容器中或者add也行，而这个Fragment中有一个listview，当我们点击这个ListView中的一项， 中间容器中的Fragment就会被replace成对应详细信息的Fragment所替代，如果我们只是replace的话， 就不会保存第一个Fragment的状态，用户又得从头开始浏览，这肯定是很不方便的，这里我们可以通过Fragment栈的addtobackStack和popbackstack来解决这个问题！当replace的同时，我们将被替换 的Fragment添加到stack中，当用户点击回退按钮时，调用popbackstack弹出栈，具体实现见下述代码 示例！

2.代码示例：简单新闻类APP列表和内容切换的实现

运行效果图：



实现代码：

Step 1:先把两个Fragment以及Activity的布局实现了

fg_newlist.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:orientation="horizontal">

    <ListView
        android:id="@+id/list_news"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

fg_context.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:textColor="@color/blue"
        android:textSize="20sp" />

</LinearLayout>
```

activity_main.xml:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txt_title"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:background="@color/blue"
        android:textColor="@color/white"
        android:text="新闻列表"
        android:textSize="20sp"
        android:textStyle="bold"
        android:gravity="center"/>

    <FrameLayout
        android:id="@+id/fl_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/txt_title"/>

</RelativeLayout>
```

Step 2:实现我们的业务Bean类和自定义BaseAdapter类 ：

Data.java:

```
/**
 * Created by Jay on 2015/9/6 0006.
 */
public class Data {

    private String new_title;
    private String new_content;

    public Data(){}

    public Data(String new_title, String new_content) {
        this.new_title = new_title;
        this.new_content = new_content;
    }

    public String getNew_title() {
        return new_title;
    }

    public String getNew_content() {
        return new_content;
    }

    public void setNew_title(String new_title) {
        this.new_title = new_title;
    }

    public void setNew_content(String new_content) {
        this.new_content = new_content;
    }
}
```

MyAdapter.java:

```
/**
 * Created by Jay on 2015/9/6 0006.
 */
public class MyAdapter extends BaseAdapter{

    private List<Data> mData;
    private Context mContext;

    public MyAdapter(List<Data> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        ViewHolder viewHolder;
        if(convertView == null){
            convertView = LayoutInflater.from(mContext).inflate(R.l
            viewHolder = new ViewHolder();
            viewHolder.txt_item_title = (TextView) convertView.find
            convertView.setTag(viewHolder);
        }else{
            viewHolder = (ViewHolder) convertView.getTag();
        }
        viewHolder.txt_item_title.setText(mData.get(position).getNe
        return convertView;
    }

    private class ViewHolder{
        TextView txt_item_title;
    }

}
```

Step 3:MainActivity的实现

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

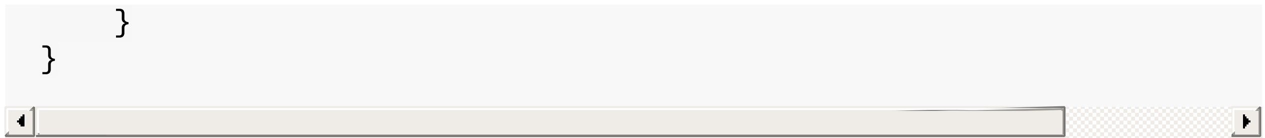
    private TextView txt_title;
    private FrameLayout fl_content;
    private Context mContext;
    private ArrayList<Data> datas = null;
    private FragmentManager fManager = null;
    private long exitTime = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        fManager = getFragmentManager();
        bindViews();

        datas = new ArrayList<Data>();
        for (int i = 1; i <= 20; i++) {
            Data data = new Data("新闻标题" + i, i + "~新闻内容~~~~~~
            datas.add(data);
        }
        NewListFragment nlFragment = new NewListFragment(fManager,
        FragmentTransaction ft = fManager.beginTransaction();
        ft.replace(R.id.fl_content, nlFragment);
        ft.commit();
    }

    private void bindViews() {
        txt_title = (TextView) findViewById(R.id.txt_title);
        fl_content = (FrameLayout) findViewById(R.id.fl_content);
    }

    //点击回退键的处理：判断Fragment栈中是否有Fragment
    //没，双击退出程序，否则像是Toast提示
    //有，popbackstack弹出栈
    @Override
    public void onBackPressed() {
        if (fManager.getBackStackEntryCount() == 0) {
            if ((System.currentTimeMillis() - exitTime) > 2000) {
                Toast.makeText(getApplicationContext(), "再按一次退出
                        Toast.LENGTH_SHORT).show();
                exitTime = System.currentTimeMillis();
            } else {
                super.onBackPressed();
            }
        } else {
            fManager.popBackStack();
            txt_title.setText("新闻列表");
        }
```



Step 4:列表Fragment的实现：

NewListFragment.java：

```
package com.jay.fragmentdemo4;

import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;

import java.util.ArrayList;

/**
 * Created by Jay on 2015/9/6 0006.
 */
public class NewListFragment extends Fragment implements AdapterVie
    private FragmentManager fManager;
    private ArrayList<Data> datas;
    private ListView list_news;

    public NewListFragment(FragmentManager fManager, ArrayList<Data
        this.fManager = fManager;
        this.datas = datas;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_newlist, containe
        list_news = (ListView) view.findViewById(R.id.list_news);
        MyAdapter myAdapter = new MyAdapter(datas, getActivity());
        list_news.setAdapter(myAdapter);
        list_news.setOnItemClickListener(this);
        return view;
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int p
        FragmentTransaction fTransaction = fManager.beginTransacti
        NewContentFragment ncFragment = new NewContentFragment();
        Bundle bd = new Bundle();
        bd.putString("content", datas.get(position).getNew_content(
```

```
            ncFragment.setArguments(bd);
            //获取Activity的控件
            TextView txt_title = (TextView) getActivity().findViewById(
            txt_title.setText(datas.get(position).getNew_content());
            //加上Fragment替换动画
            fTransaction.setCustomAnimations(R.anim.fragment_slide_left
            fTransaction.replace(R.id.fl_content, ncFragment);
            //调用addToBackStack将Fragment添加到栈中
            fTransaction.addToBackStack(null);
            fTransaction.commit();
        }
}
```

Step 5:内容Fragment的实现：

NewContentFragment.java:

```
/**
 * Created by Jay on 2015/9/6 0006.
 */
public class NewContentFragment extends Fragment {

    NewContentFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup cor
        View view = inflater.inflate(R.layout.fg_content, container
        TextView txt_content = (TextView) view.findViewById(R.id.tx
        //getArgument获取传递过来的Bundle对象
        txt_content.setText(getArguments().getString("content"));
        return view;
    }

}
```

代码很简单，就不慢慢解释了~

3.代码下载

FragmentDemo5.zip：下载 FragmentDemo5.zip

本节小结：

> 因为时间的关系，并没有详细的去做过多的讲解，示例代码也很简单，方便各位初学者理解！ 如果要用到实际项目中还需要对此进行一番修改~！好的，本节就到这里，谢谢~

6.1 数据存储与访问之——文件存储读写

本节引言：

> 嘿嘿，看到这个题目，相信部分读者会问，你前面的Fragment写完了吗？嗯，没写完，因为想例子，需要 一点时间，为了提高效率，所以决定像多线程一样，并发的来写教程，这样可能可以加快写教程的进度， 到现在为止，刚好写了60篇，离完成入门教程还很远呢，而前面也说过，想在一个半到两个月之内完成 这套教程，今天已经9.1号了，要加吧劲~好的，废话就这么多，本节给大家介绍的是Android数据存储与 访问方式中的一个——文件存储与读写，当然除了这种方式外，我们可以存到SharedPreference，数据库， 或者Application中，当然这些后面都会讲，嗯，开始本节内容~

1.Android文件的操作模式

> 学过Java的同学都知道，我们新建文件，然后就可以写入数据了，但是Android却不一样，因为Android是 基于Linux的，我们在读写文件的时候，还需加上文件的操作模式，Android中的操作模式如下：

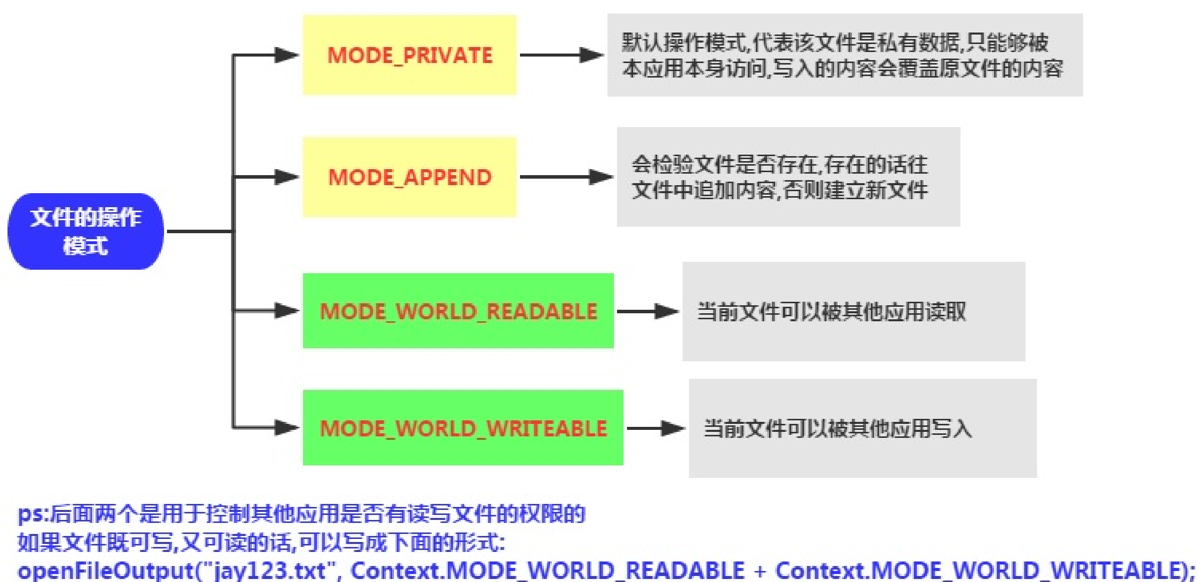


图1：文件的操作模式

2.文件的相关操作方法

openFileOutput(filename,mode)	打开文件输出流,就是往文件中写入数据,第二个参数是模式
openFileIntput(filename)	打开文件输入流,就是读取文件中的信息到程序中
getDir(name,mode)	在app的data目录下获取或创建name对应的子目录
getFileDir()	获得app的data目录的file目录的绝对路径
String[] fileList()	返回app的data目录下的全部文件
deleteFile(filename)	删除app的data目录下的指定文件

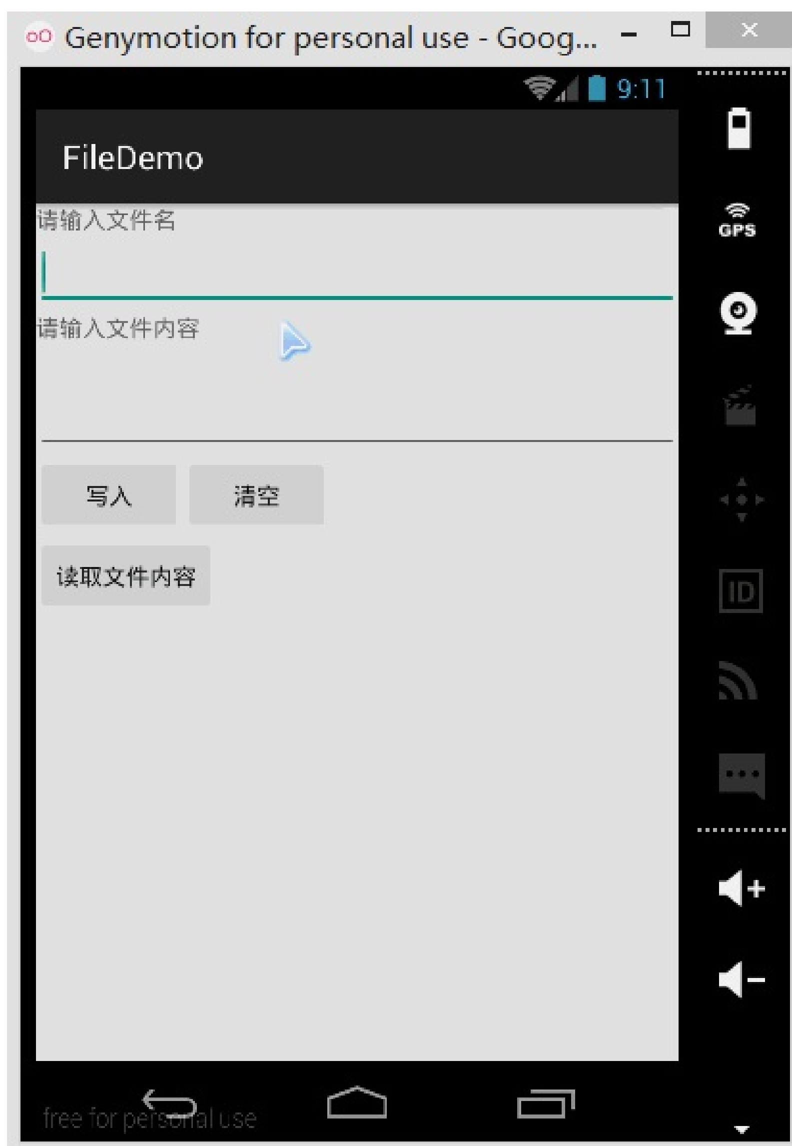
ps:Android有自己的一套安全模型,当我们安装apk时,系统会分配给他一个userid,当该应用要去访问其他资源,比如文件时就需要匹配userid,任何app创建的文件,sharedpreferences,数据库文件都是私有的,其他程序无法访问;除非在创建时指定了Context.MODE_WORLD_READABLE或者Context.MODE_WORLD_WRITEABLE ，只有这样其他程序才能正确访问。

图2：文件的相关操作方法

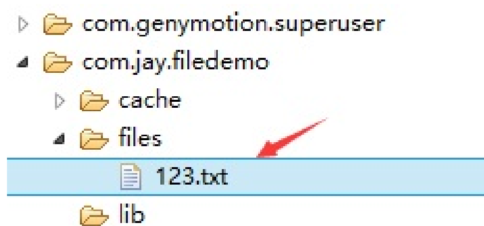
3.文件读写的实现

> Android中的文件读写和Java中的文件I/O相同，流程也很简单，下面我们来写个简单的示例：

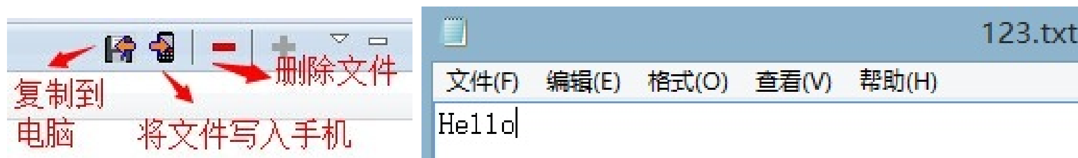
实现效果图：



PS:这里用的是模拟器，因为笔者的N5并没有root，看不到文件的存储目录，下面我们打开DDMS 的File Explorer可以看到，在data/data/<包名>/file中有我们写入的文件：



我们可以点击右上角的响应图标将文件导入到电脑中，并且打开验证写入的内容：



代码实现：

首先是布局文件:**main_activity.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.jay.example.filedemo1.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nametitle" />

    <EditText
        android:id="@+id/editname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/detailtitle" />

    <EditText
        android:id="@+id/editdetail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minLines="2" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnsave"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btnwrite" />

        <Button
            android:id="@+id/btnclean"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btnclean" />
    </LinearLayout>

    <Button
        android:id="@+id/btnread"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="@string/btnread" />

</LinearLayout>
```

然后我们来写一个文件协助类：**FileHelper.java**

```
/**
 * Created by Jay on 2015/9/1 0001.
 */
public class FileHelper {

    private Context mContext;

    public FileHelper() {
    }

    public FileHelper(Context mContext) {
        super();
        this.mContext = mContext;
    }

    /*
    * 这里定义的是一个文件保存的方法，写入到文件中，所以是输出流
    * */
    public void save(String filename, String filecontent) throws Ex
        //这里我们使用私有模式,创建出来的文件只能被本应用访问,还会覆盖原文件哦
        FileOutputStream output = mContext.openFileOutput(filename,
        output.write(filecontent.getBytes());  //将String字符串以字节
        output.close();         //关闭输出流
    }

    /*
    * 这里定义的是文件读取的方法
    * */
    public String read(String filename) throws IOException {
        //打开文件输入流
        FileInputStream input = mContext.openFileInput(filename);
        byte[] temp = new byte[1024];
        StringBuilder sb = new StringBuilder("");
        int len = 0;
        //读取文件内容:
        while ((len = input.read(temp)) > 0) {
            sb.append(new String(temp, 0, len));
        }
        //关闭输入流
        input.close();
        return sb.toString();
    }

}
```

最后是**MainActivity.java**，我们在这里完成相关操作：

```
public class MainActivity extends AppCompatActivity implements View

    private EditText editname;
```

```
    private EditText editdetail;
    private Button btnsave;
    private Button btnclean;
    private Button btnread;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = getApplicationContext();
        bindViews();
    }

    private void bindViews() {
        editdetail = (EditText) findViewById(R.id.editdetail);
        editname = (EditText) findViewById(R.id.editname);
        btnclean = (Button) findViewById(R.id.btnclean);
        btnsave = (Button) findViewById(R.id.btnsave);
        btnread = (Button) findViewById(R.id.btnread);

        btnclean.setOnClickListener(this);
        btnsave.setOnClickListener(this);
        btnread.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btnclean:
                editdetail.setText("");
                editname.setText("");
                break;
            case R.id.btnsave:
                FileHelper fHelper = new FileHelper(mContext);
                String filename = editname.getText().toString();
                String filedetail = editdetail.getText().toString(
                try {
                    fHelper.save(filename, filedetail);
                    Toast.makeText(getApplicationContext(), "数据写
                } catch (Exception e) {
                    e.printStackTrace();
                    Toast.makeText(getApplicationContext(), "数据写
                }
                break;
            case R.id.btnread:
                String detail = "";
                FileHelper fHelper2 = new FileHelper(getApplicatio
                try {
                    String fname = editname.getText().toString();
                    detail = fHelper2.read(fname);
                } catch (IOException e) {
                    e.printStackTrace();
```

```
                        }
                        Toast.makeText(getApplicationContext(), detail, Toa
                        break;
                }
            }
    }
```

4.读取SD卡上的文件

读取流程图：



代码示例：

运行效果图：



同样打开DDMS的File Explorer，在旧版本的系统上我们可以直接在mmt\sdcard上找到，但是新版本 的就可能需要我们自己找找了，首先我们来到这个路径下：

Name	Date	Time	Permissions	Info
mnt	2015-09-01	09:06	drwxrwxr-x	
USB	2015-09-01	09:06	d---------	
asec	2015-09-01	09:06	drwxr-xr-x	
obb	2015-09-01	09:06	drwxr-xr-x	
sdcard	2015-09-01	09:06	lrwxrwxrwx	-> /storage/emulated/legacy

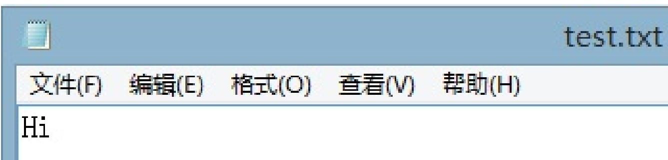
点开sdcard，但是没东西，我们继续找唠叨后面这个/storage/emulated/legacy下找：

Name	Date	Time	Permissions	Info
storage	2015-09-01	09:06	d---r-x---	
emulated	2015-09-01	09:06	dr-xr-xr-x	
legacy	2015-09-01	09:06	lrwxrwxrwx	-> /mnt/shell/emulated/0
sdcard0	2015-09-01	09:06	lrwxrwxrwx	-> /storage/emulated/legacy

好吧，他又跳到别的地方去了，我们继续找/storage/shell/emilated/0

Name	Size	Date	Time	Permissions
shell		2015-09-01	09:06	drwx------
emulated		1970-01-01	00:00	drwxrwxrwx
0		2015-09-01	11:05	drwxrwxrwx
Alarms		2015-08-04	02:17	drwxrwxrwx
Android		2015-08-27	07:18	drwxrwxrwx
DCIM		2015-08-04	02:17	drwxrwxrwx
Download		2015-08-27	07:16	drwxrwxrwx
Movies		2015-08-04	02:17	drwxrwxrwx
Music		2015-08-04	02:17	drwxrwxrwx
Notifications		2015-08-04	02:17	drwxrwxrwx
Pictures		2015-08-27	07:18	drwxrwxrwx
Podcasts		2015-08-04	02:17	drwxrwxrwx
Ringtones		2015-08-04	02:17	drwxrwxrwx
Tencent		2015-08-27	07:18	drwxrwxrwx
baidu		2015-08-27	07:19	drwxrwxrwx
test.txt	2	2015-09-01	11:05	-rwxrwxrwx
xtuone		2015-08-27	07:18	drwxrwxrwx
legacy		2015-08-04	02:17	drwxrwxrwx
obb		2015-08-04	02:17	drwxrwxrwx

果然找到了，我们在SD卡里生成的test.txt！导出到电脑看下里面的内容：



嘿嘿，果然读写SD卡成功~接下来我们来看下代码是怎么写的：

代码实现：

main_activity.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.jay.example.filedemo2.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="清输入文件名" />

    <EditText
        android:id="@+id/edittitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="文件名" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="清输入文件内容" />

    <EditText
        android:id="@+id/editdetail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="文件内容" />

    <Button
        android:id="@+id/btnsave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="保存到SD卡" />

    <Button
        android:id="@+id/btnclean"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="清空" />

    <Button
        android:id="@+id/btnread"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="读取sd卡中的文件" />

</LinearLayout>
```

接着我们来写一个SD操作类： **SDFileHelper.java**

```
/**
 * Created by Jay on 2015/9/1 0001.
 */
public class SDFileHelper {

    private Context context;

    public SDFileHelper() {
    }

    public SDFileHelper(Context context) {
        super();
        this.context = context;
    }

    //往SD卡写入文件的方法
    public void savaFileToSD(String filename, String filecontent) t
        //如果手机已插入sd卡,且app具有读写sd卡的权限
        if (Environment.getExternalStorageState().equals(Environmer
            filename = Environment.getExternalStorageDirectory().ge
            //这里就不要用openFileOutput了,那个是往手机内存中写数据的
            FileOutputStream output = new FileOutputStream(filename
            output.write(filecontent.getBytes());
            //将String字符串以字节流的形式写入到输出流中
            output.close();
            //关闭输出流
        } else Toast.makeText(context, "SD卡不存在或者不可读写", Toast
    }

    //读取SD卡中文件的方法
    //定义读取文件的方法:
    public String readFromSD(String filename) throws IOException {
        StringBuilder sb = new StringBuilder("");
        if (Environment.getExternalStorageState().equals(Environmer
            filename = Environment.getExternalStorageDirectory().ge
            //打开文件输入流
            FileInputStream input = new FileInputStream(filename);
            byte[] temp = new byte[1024];

            int len = 0;
            //读取文件内容:
            while ((len = input.read(temp)) > 0) {
                sb.append(new String(temp, 0, len));
            }
            //关闭输入流
            input.close();
        }
        return sb.toString();
    }
```

```
}
```

接着**MainActivity.java**实现相关逻辑：

```
public class MainActivity extends AppCompatActivity implements View

    private EditText editname;
    private EditText editdetail;
    private Button btnsave;
    private Button btnclean;
    private Button btnread;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = getApplicationContext();
        bindViews();
    }

    private void bindViews() {
        editname = (EditText) findViewById(R.id.edittitle);
        editdetail = (EditText) findViewById(R.id.editdetail);
        btnsave = (Button) findViewById(R.id.btnsave);
        btnclean = (Button) findViewById(R.id.btnclean);
        btnread = (Button) findViewById(R.id.btnread);

        btnsave.setOnClickListener(this);
        btnclean.setOnClickListener(this);
        btnread.setOnClickListener(this);
    }

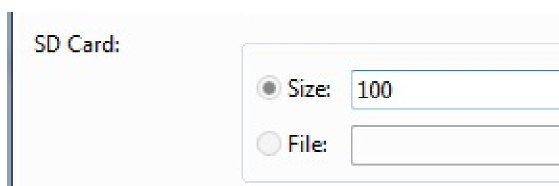
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btnclean:
                editdetail.setText("");
                editname.setText("");
                break;
            case R.id.btnsave:
                String filename = editname.getText().toString();
                String filedetail = editdetail.getText().toString(
                SDFileHelper sdHelper = new SDFileHelper(mContext)
                try
                {
                    sdHelper.savaFileToSD(filename, filedetail);
                    Toast.makeText(getApplicationContext(), "数据写
                }
                catch(Exception e){
```

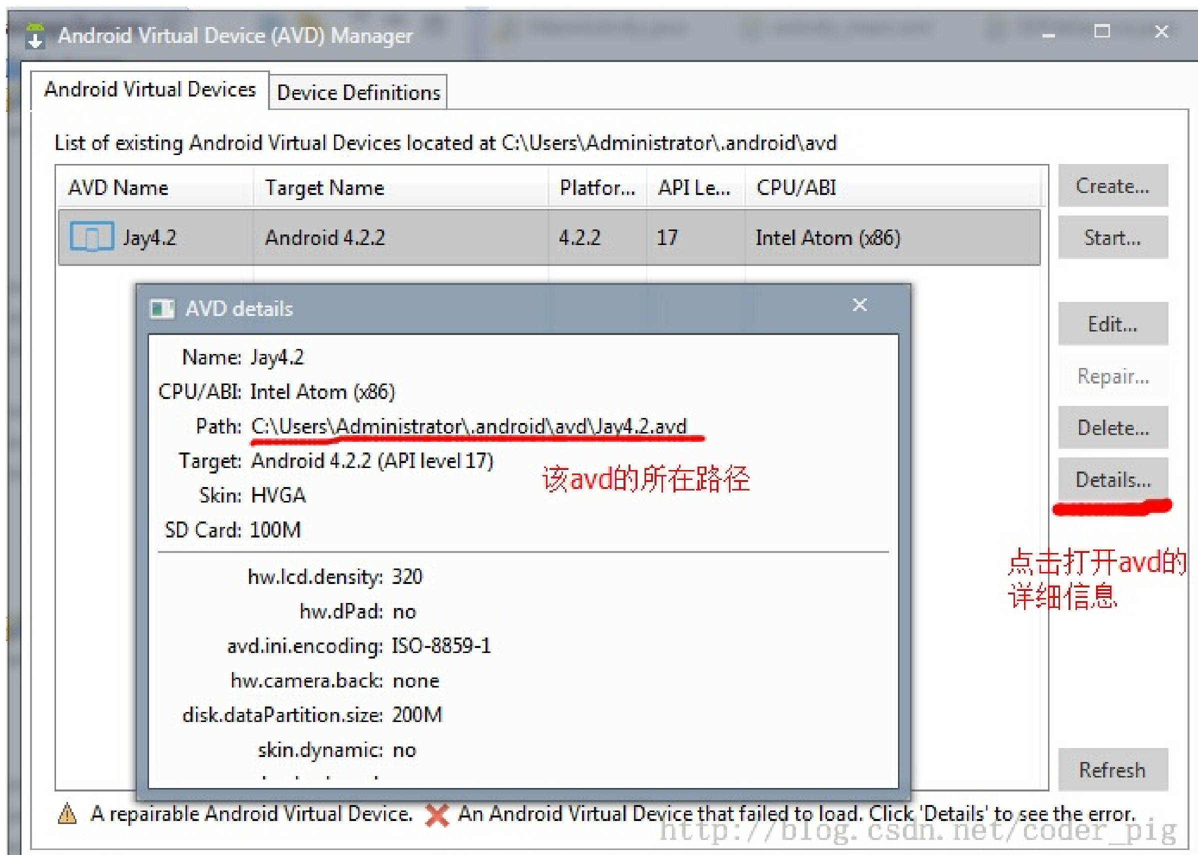
```
                    e.printStackTrace();
                    Toast.makeText(getApplicationContext(), "数据写入
                }
                break;
            case R.id.btnread:
                String detail = "";
                SDFileHelper sdHelper2 = new SDFileHelper(mContext);
                try
                {
                    String filename2 = editname.getText().toString(
                    detail = sdHelper2.readFromSD(filename2);
                }
                catch(IOException e){e.printStackTrace();}
                Toast.makeText(getApplicationContext(), detail, Toa
                break;
        }
    }
}
```

最后别忘记在**AndroidManifest.xml**写上读写SD卡的权限哦！

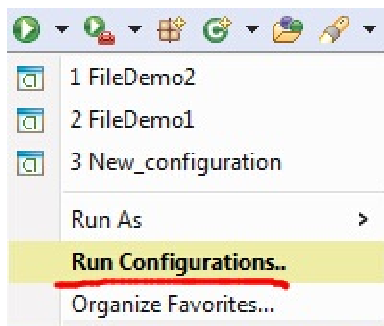
```
<!-- 在SDCard中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FIL
<!-- 往SDCard写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_ST
```

5.关于原生模拟器SD卡的问题

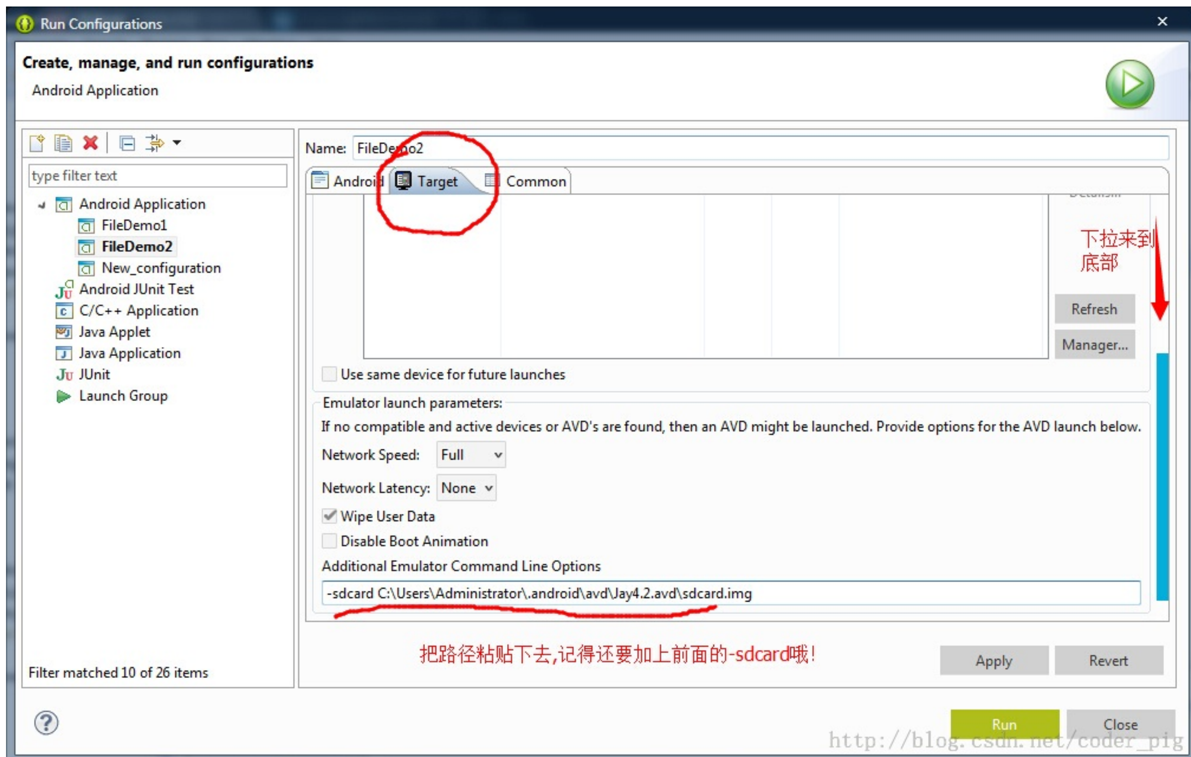
> 如果是真机调试的话通常都是可以的,对于原生虚拟机的话就问题多多了,再我们前面使用Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)可能 一直返回的是false，就是SD卡不存在，这个是主要的问题，现在新版本的SDK都会在 创建AVD的时候会同时申请一块SD卡的存储区域的
>
> 
>
>
> 对于旧版本的sdk或者其他原因可能需要手动关联下sd卡,设置如下: ①找到创建好的avd的镜像的路径: 点击打开avd界面,点击detail,查看avd镜像的目录下



②来到avd镜像所在的路径下,复制sdcard.img的路径: 比如我的:-sdcard C:\Users\Administrator.android\avd\Jay4.2.avd\sdcard.img



③接着点击 来到以下界面:



> 最后apply以下,然后Run就可以了!

6.读取raw和assets文件夹下的文件

> 相信大家对两个文件夹并不陌生，如果我们不想自己的文件被编译成二进制文件的话， 我们可以把文件放到这两个目录下，而两者的区别如下：
>
> - **res/raw**：文件会被映射到R.java文件中，访问的时候直接通过资源ID即可访问，而且 他不能有目录结构，就是不能再创建文件夹
> - **assets**：不会映射到R.java文件中，通过AssetManager来访问，能有目录结构，即， 可以自行创建文件夹

读取文件资源：

res/raw：

```
InputStream is =getResources().openRawResource(R.raw.filename);
```

assets：

```
AssetManager am =  getAssets();
InputStream is = am.open("filename");
```

代码下载：

- **FileDemo.zip**：下载 **FileDemo.zip**
- **FileDemo2.zip**：下载 **FileDemo2.zip**

本节小结：

> 好的，关于Android的数据存储与访问的第一节——文件读写就到这里，如果在学习本文中 遇到什么问题，或者觉得有些纰漏的地方，欢迎提出，万分感激，谢谢~

6.2 数据存储与访问之——SharedPreferences保存用户偏好参数

本节引言：

> 本节给大家介绍的是第二种存储用户数据的方式，使用SharedPreferences(保存用户偏好参数)保存数据， 当我们的应用想要保存用户的一些偏好参数，比如是否自动登陆，是否记住账号密码,是否在Wifi下才能 联网等相关信息,如果使用数据库的话,显得有点大材小用了！我们把上面这些配置信息称为用户的偏好设置，就是用户偏好的设置，而这些配置信息通常是保存在特定的文件中！比如windows使用ini文件， 而J2SE中使用properties属性文件与xml文件来保存软件的配置信息;而在Android中我们通常使用 一个轻量级的存储类——SharedPreferences来保存用户偏好的参数！SharedPreferences也是使用xml文件, 然后类似于Map集合,使用键-值的形式来存储数据;我们只需要调用SharedPreferences的getXxx(name), 就可以根据键获得对应的值！使用起来很方便！

1.SharedPreferences使用示例：

使用流程图：

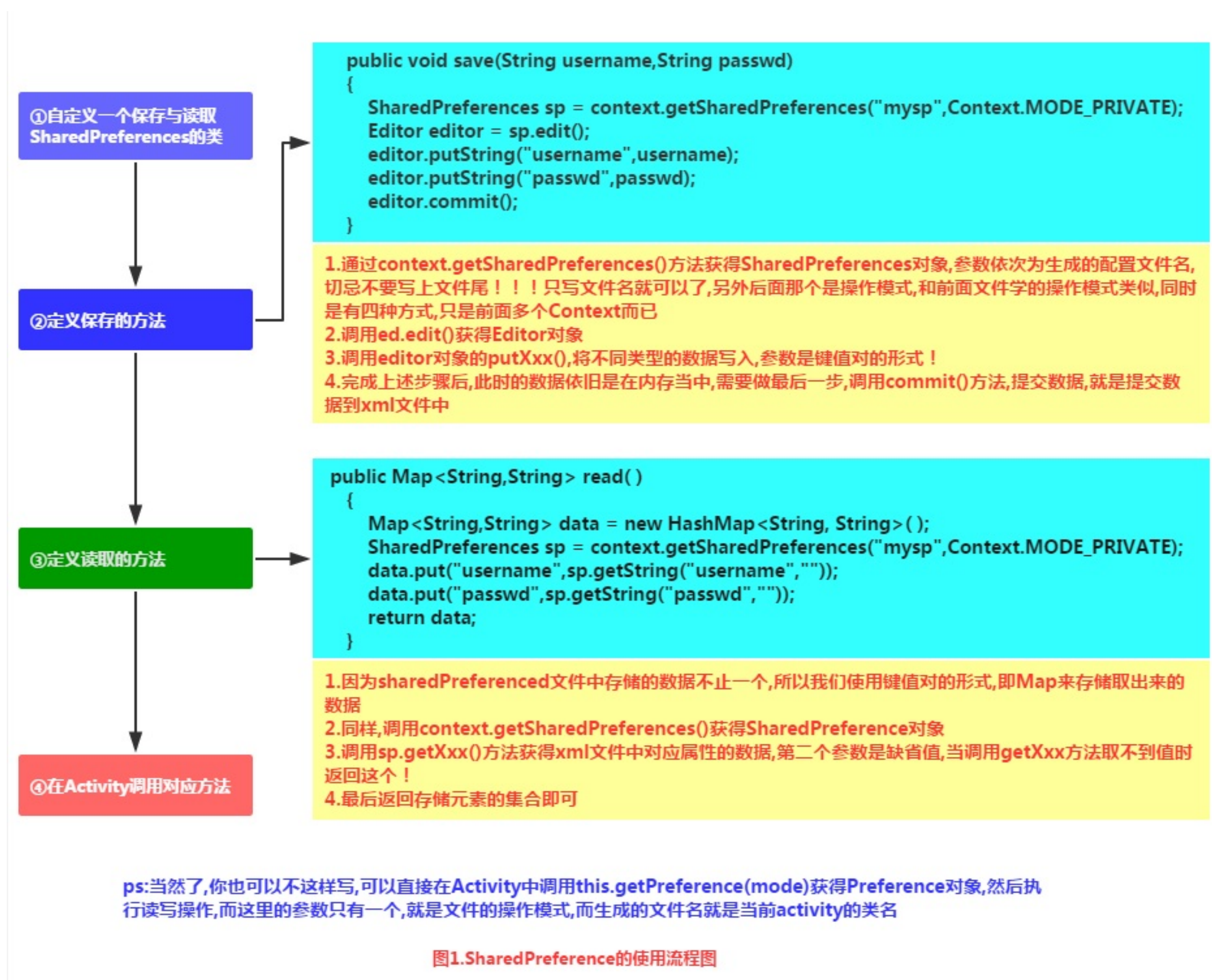


图1.SharedPreference的使用流程图

实现代码示例：

运行效果图：

流程是输入账号密码后点击登录,将信息保存到SharedPreference文件中, 然后重启app,看到数据已经显示在文本框中了



另外保存后，我们可以在File Expoler打开data/data/<包名>可以看到在shared_prefs目录下 生成了一个xml文件(因为N5没root，这里找了以前的效果图)：

Name	Size	Date	Time	Permissions
com.jay.example.sharedpreferencedemo1		2014-08-18	04:02	drwxr-x--x
cache		2014-08-18	03:52	drwxrwx--x
lib		2014-08-18	04:01	lrwxrwxrwx
shared_prefs		2014-08-18	04:02	drwxrwx--x
mysp.xml	143	2014-08-18	04:02	-rw-rw----

点击导出到桌面可以看到里面的内容:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="passwd">1234</string>
<string name="username">Jay</string>
</map>
```

代码实现：

布局文件**activity_main.xml**的编写：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MyActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="用户登陆" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="请输入用户名" />

    <EditText
        android:id="@+id/editname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="用户名" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入密码" />

    <EditText
        android:id="@+id/editpasswd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="密码"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/btnlogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="登录" />
</LinearLayout>
```

编写简单的SP工具类：**SharedHelper.java**：

```
/**
 * Created by Jay on 2015/9/2 0002.
 */
public class SharedHelper {

    private Context mContext;

    public SharedHelper() {
    }

    public SharedHelper(Context mContext) {
        this.mContext = mContext;
    }

    //定义一个保存数据的方法
    public void save(String username, String passwd) {
        SharedPreferences sp = mContext.getSharedPreferences("mysp'
        SharedPreferences.Editor editor = sp.edit();
        editor.putString("username", username);
        editor.putString("passwd", passwd);
        editor.commit();
        Toast.makeText(mContext, "信息已写入SharedPreference中", Toas
    }

    //定义一个读取SP文件的方法
    public Map<String, String> read() {
        Map<String, String> data = new HashMap<String, String>();
        SharedPreferences sp = mContext.getSharedPreferences("mysp'
        data.put("username", sp.getString("username", ""));
        data.put("passwd", sp.getString("passwd", ""));
        return data;
    }
}
```

最后是**MainActivity.java**实现相关逻辑：

```
public class MainActivity extends AppCompatActivity {

    private EditText editname;
    private EditText editpasswd;
    private Button btnlogin;
    private String strname;
    private String strpasswd;
    private SharedHelper sh;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = getApplicationContext();
        sh = new SharedHelper(mContext);
        bindViews();
    }

    private void bindViews() {
        editname = (EditText)findViewById(R.id.editname);
        editpasswd = (EditText)findViewById(R.id.editpasswd);
        btnlogin = (Button)findViewById(R.id.btnlogin);
        btnlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                strname = editname.getText().toString();
                strpasswd = editpasswd.getText().toString();
                sh.save(strname,strpasswd);
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        Map<String,String> data = sh.read();
        editname.setText(data.get("username"));
        editpasswd.setText(data.get("passwd"));
    }
}
```

2.读取其他应用的SharedPreferences

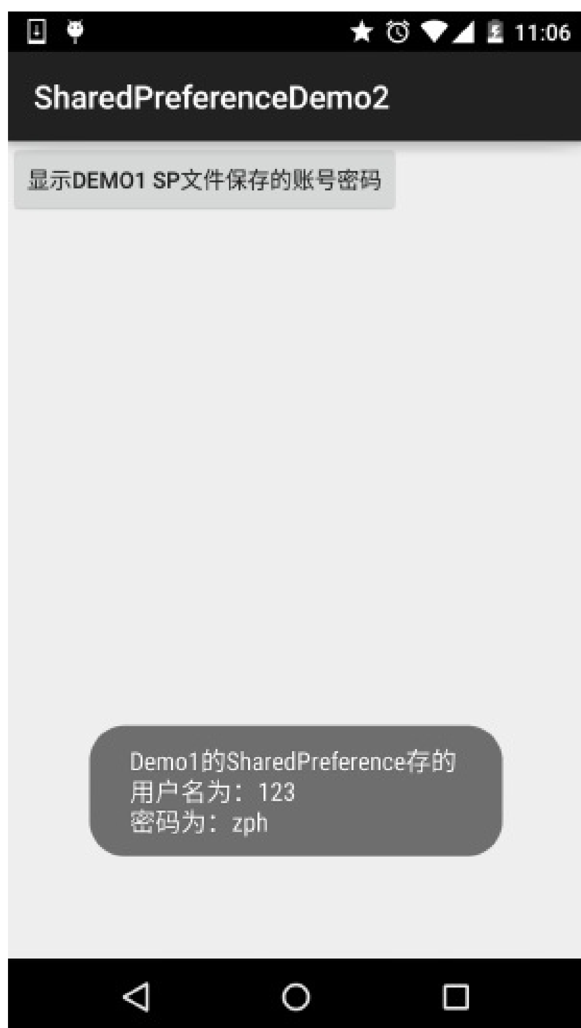
> 核心： 获得其他app的Context,而这个Context代表访问该app的全局信息的接口,而决定应用的唯一标识 是应用的包名,所以我们可以通过应用包名获得对应app的Context 另外有一点要注意的是：其他应用的SP文件是否能被读写的前提就是SP文件是否指定了可读或者 可写的权限, 我们上面创建的是MODE_PRIVATE的就不可以了~所以说你像读别人的SP里的数据, 很难, 另外, 一些关键的信息, 比如密码保存到SP里, 一般都是会做加密的, 所以只能自己写自己玩~ 等下会讲下常用的MD5加密方法！

实现流程图：



代码示例：

运行效果图：



代码实现：

我们读取SP的操作放在MainActivity.java中完成，点击按钮后读取SP，并通过Toast显示出来：

```
public class MainActivity extends AppCompatActivity {

    private Context othercontext;
    private SharedPreferences sp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnshow = (Button) findViewById(R.id.btnshow);
        btnshow.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //获得第一个应用的包名,从而获得对应的Context,需要对异常进行
                try {
                    othercontext = createPackageContext("com.jay.sh
                } catch (PackageManager.NameNotFoundException e) {
                    e.printStackTrace();
                }
                //根据Context取得对应的SharedPreferences
                sp = othercontext.getSharedPreferences("mysp", Cont
                String name = sp.getString("username", "");
                String passwd = sp.getString("passwd", "");
                Toast.makeText(getApplicationContext(), "Demo1的Sha
            }
        });
    }
}
```

3.使用MD5对SharedPreference的重要数据进行加密

> 嘿嘿，上面我们这样直接把账号密码保存到sp里，如果没root的手机，别的应用倒无法访问手机， 如果root了，然后数据给其他应用获取到，然后造成了一些后果，这...就不怪我们了，哈哈， 谁叫你root了~，这锅我们不背，的确是这样！但是作为一名有责任心的APP开发人员，我们总不能 这样是吧，我们可以使用一些加密算法对用户密码进行加密，另外我们一般加密的都是用户密码！下面我们简画个简单的图帮助大家理解下加密的处理的流程：

1.简单的加密处理流程

流程图如下：

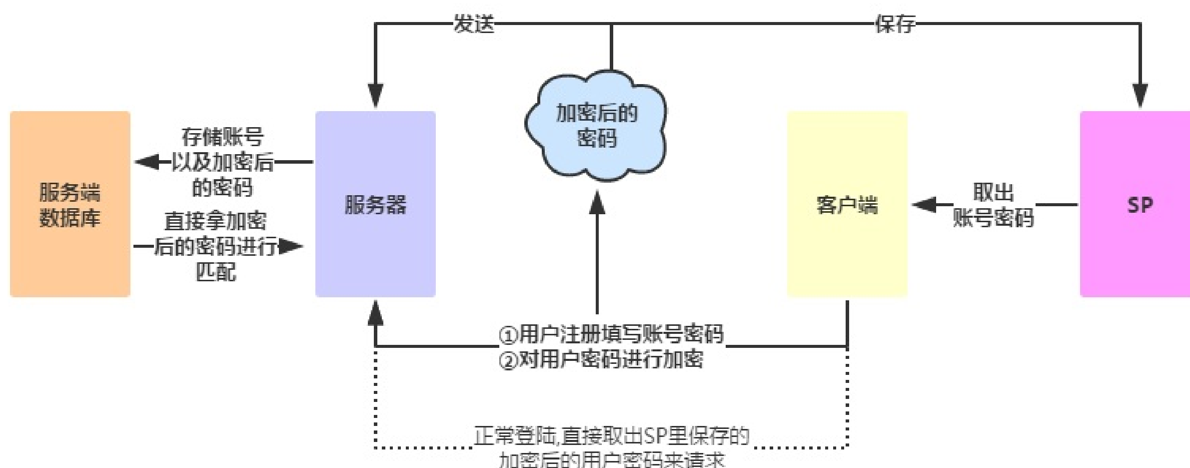


图2.简单的加密处理流程图

流程图解析：

> - **Step 1**.用户注册账号密码，账号密码校验后(账号是否重复，密码位数 > 6位等)， 即账号密码有效，注册成功后，我们提交给服务器的账号，以及本地加密过的密码！
> - **Step 2**.服务器端将用户提交的账号，加密过的密码保存到服务端的数据库中，也就是服务 端并不会保存我们的明文密码(原始)密码！
> - **Step 3**.说回客户端，如果注册成功或者登陆成功，你想保存账号密码到SP中，保存的的密码 也需要走一趟加密流程！即明文密码——>加密，再保存！如果不保存，每次请求的时候，明文密码 也要走一趟家里流程，然后拿着加密后的密码来请求服务器！
> - **Step 4**.服务器验证账号以及加密密码，成功，分配客户端一个session标识，后续客户端可以拿着 这个session来访问服务端提供的相关服务！

嘿嘿，理解了吧，加密的方法有很多种，小猪也不是这方面的高玩，以前使用过的加密方法是MD5 加密，本节也给大家简单介绍一下这个MD5加密，以及演示下用法~

2.MD5简单介绍：

1）MD5是什么鬼？：

> 答：Message Digest Algorithm MD5（中文名为消息摘要算法第五版）为计算机安全领域广泛 使用的一种散列函数，用以提供消息的完整性保护——摘自《百度百科》 简单点说就是一种加密算法，可以将一个字符串，或者文件，压缩包，执行MD5加密后， 就可以生产一个固定长度为128bit的串！这个串基本唯一！另外我们都知道:一个十六进制 需要用4个bit来表示，那么对应的MD5的字符串长度就为：128 / 4 = 32位了！另外可能 你看到一些md5是16位的，只是将32位MD5码去掉了前八位以及后八位！不信么，我们来试试 百度一下：md5在线解密，第一个：http://www.cmd5.com/
>
> 密文: 曹神带我日狗
> 类型: md5 [帮助]
> 解密
>
> 查询结果:
> md5(曹神带我日狗,32) = 32a16e933ea4bc45f34790b015012ec0
> md5(曹神带我日狗,16) = 3ea4bc45f34790b0

2）MD5能破解吗？

> 答：MD5不可逆，就是说没有对应的算法，无法从生成的md5值逆向得到原始数据！ 当然暴力破解除外， 简单的MD5加密后可以查MD5库~

3）MD5值唯一吗？

> 答：不唯一，一个原始数据只对应一个MD5值，但是一个MD5值可能对应多个原始数据！

3.MD5加密实现例子：

其实网上有很多写好的MD5的例子，百度或者谷歌一搜一大堆，这里提供下小猪用的MD5加密工具类！

Md5Util.java：

```
/**
 * Created by Jay on 2015/9/2 0002.
 */
public class MD5 {
    public static String getMD5(String content) {
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5"
            digest.update(content.getBytes());
            return getHashString(digest);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return null;
    }

    private static String getHashString(MessageDigest digest) {
        StringBuilder builder = new StringBuilder();
        for (byte b : digest.digest()) {
            builder.append(Integer.toHexString((b >> 4) & 0xf));
            builder.append(Integer.toHexString(b & 0xf));
        }
        return builder.toString();
    }
}
```

MainActivity.java直接调用getMD5这个静态方法：

```
Log.e("HeHe", MD5.getMD5("呵呵"));
```

我们可以看到Logcat上打印出：

E/HeHe： 86d51ce7753a36079041fc15f7248035

这就是加密过后的呵呵了，我们可以把这串密文拷贝到上面这个md5的在线解密网站：

密文: 86d51ce7753a36079041fc15f7248035
类型: md5 [帮助]
解密

查询结果:
呵呵

[添加备注]

嘿嘿，果然，只是这样加密一次，就直接破解了，有点不安全的样子，那就加密100次咯， 就是将加密后的字符串再加密，重复100次，我们在原先的基础上加个加密一百次的方法：

```
public static String getMD5x100(String content){
    String s1 = content;
    for(int i = 0;i < 100;i++){
        s1 = getMD5(s1);
    }
    return s1;
}
```

然后调用下，发现打印这个的Log：

```
E/HeHe :  50db93f6f9a34e73ae5d1e6953601941
```

复制界面网站上：

密文: 50db93f6f9a34e73ae5d1e6953601941
类型: md5 [帮助]
解密

查询结果:
未查到,已加入本站后台解密，完成进度:2%
请注册并登录查询，若解密成功自动给你发送邮件通知。

[添加备注]

好的，装B成功~

4.SharedPreference工具类：

> 每次都要自行实例化SP相关的类，肯定很麻烦，这里贴个SP的工具类，大家可以贴到 自己的项目中，工具类来源于鸿洋大神的blog~

SPUtils.java

```
package com.jay.sharedpreferencedemo3;

import android.content.Context;
import android.content.SharedPreferences;

import java.util.Map;

/**
```

```
 * Created by Jay on 2015/9/2 0002.
 */
public class SPUtils {
    /**
     * 保存在手机里的SP文件名
     */
    public static final String FILE_NAME = "my_sp";

    /**
     * 保存数据
     */
    public static void put(Context context, String key, Object obj)
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        SharedPreferences.Editor editor = sp.edit();
        if (obj instanceof Boolean) {
            editor.putBoolean(key, (Boolean) obj);
        } else if (obj instanceof Float) {
            editor.putFloat(key, (Float) obj);
        } else if (obj instanceof Integer) {
            editor.putInt(key, (Integer) obj);
        } else if (obj instanceof Long) {
            editor.putLong(key, (Long) obj);
        } else {
            editor.putString(key, (String) obj);
        }
        editor.commit();
    }

    /**
     * 获取指定数据
     */
    public static Object get(Context context, String key, Object de
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        if (defaultObj instanceof Boolean) {
            return sp.getBoolean(key, (Boolean) defaultObj);
        } else if (defaultObj instanceof Float) {
            return sp.getFloat(key, (Float) defaultObj);
        } else if (defaultObj instanceof Integer) {
            return sp.getInt(key, (Integer) defaultObj);
        } else if (defaultObj instanceof Long) {
            return sp.getLong(key, (Long) defaultObj);
        } else if (defaultObj instanceof String) {
            return sp.getString(key, (String) defaultObj);
        }
        return null;
    }

    /**
     * 删除指定数据
     */
    public static void remove(Context context, String key) {
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        SharedPreferences.Editor editor = sp.edit();
```

```
        editor.remove(key);
        editor.commit();
    }

    /**
     * 返回所有键值对
     */
    public static Map<String, ?> getAll(Context context) {
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        Map<String, ?> map = sp.getAll();
        return map;
    }

    /**
     * 删除所有数据
     */
    public static void clear(Context context) {
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        SharedPreferences.Editor editor = sp.edit();
        editor.clear();
        editor.commit();
    }

    /**
     * 检查key对应的数据是否存在
     */
    public static boolean contains(Context context, String key) {
        SharedPreferences sp = context.getSharedPreferences(FILE_NA
        return sp.contains(key);
    }

}
```

5.代码下载：

SharedPreferenceDemo.zip：下载 SharedPreferenceDemo.zip
SharedPreferenceDemo2.zip：下载 SharedPreferenceDemo2.zip
SharedPreferenceDemo3.zip：下载 SharedPreferenceDemo3.zip

本节小结：

> 好的，关于Android存储数据的第二种方式：SharedPreference保存用户偏好参数的内容就这么多， 应该可以满足你日常开发使用SP的需求，如果有什么遗漏，欢迎提出，谢谢~

6.3.1 数据存储与访问之——初见SQLite数据库

本节引言：

> 本节我们继续来学习Android数据存储与访问的第三种方式：SQLite数据库，和其他的SQL数据库不同， 我们并不需要在手机上另外安装一个数据库软件，Android系统已经集成了这个数据库，我们无需像 使用其他数据库软件(Oracle，MSSQL，MySql等)又要安装，然后完成相关配置，又要改端口之类的！ 引言就说这么多，接下来我们来学习下这个东西~

1.基本概念

1）SQLite是什么？为什么要用SQLite？SQLite有什么特点？

> 答：下面请听小猪娓娓道来：
>
> ①SQLite是一个轻量级的关系型数据库，运算速度快，占用资源少，很适合在移动设备上使用， 不仅支持标准SQL语法，还遵循ACID(数据库事务)原则，无需账号，使用起来非常方便！
>
> ②前面我们学习了使用文件与SharedPreference来保存数据,但是在很多情况下， 文件并不一定是有效的,如多线程并发访问是相关的；app要处理可能变化的复杂数据结构等等！ 比如银行的存钱与取钱！使用前两者就会显得很无力或者繁琐，数据库的出现可以解决这种问题， 而Android又给我们提供了这样一个轻量级的SQLite，为何不用？
>
> ③SQLite支持五种数据类型:NULL,INTEGER,REAL(浮点数),TEXT(字符串文本)和BLOB(二进制对象) 虽然只有五种,但是对于varchar,char等其他数据类型都是可以保存的;因为SQLite有个最大的特点: 你可以各种数据类型的数据保存到任何字段中而不用关心字段声明的数据类型是什么,比如你 可以在Integer类型的字段中存放字符串,当然除了声明为主键**INTEGER PRIMARY KEY**的字段只能够存储**64**位整数！ 另外， SQLite 在解析CREATE TABLE 语句时， 会忽略 CREATE TABLE 语句中跟在字段名后面的数据类型信息如下面语句会忽略 name字段的类型信息： **CREATE TABLE person (personid integer primary key autoincrement, name varchar(20))**
>
> 小结下特点：
>
> SQlite通过文件来保存数据库，一个文件就是一个数据库，数据库中又包含多个表格，表格里又有 多条记录，每个记录由多个字段构成，每个字段有对应的值，每个值我们可以指定类型，也可以不指定 类型(主键除外)

PS：对了，Android内置的SQLite是SQLite 3版本的~

2）几个相关的类：

> 嘿嘿，学习一些新东西的时候，最不喜欢的莫过于遇到一些新名词，是吧，我们先来说下几个 我们在使用数据库时用到的三个类：
>
> - **SQLiteOpenHelper**：抽象类，我们通过继承该类，然后重写数据库创建以及更新的方法， 我们还可以通过该类的对象获得数据库实例，或者关闭数据库！
> - **SQLiteDatabase**：数据库访问类：我们可以通过该类的对象来对数据库做一些增删改查的操作
> - **Cursor**：游标，有点类似于JDBC里的resultset，结果集！可以简单理解为指向数据库中某 一个记录的指针！

2.使用SQLiteOpenHelper类创建数据库与版本管理

> 对于涉及数据库的app,我们不可能手动地去给他创建数据库文件,所以需要在第一次启用app 的时候就创建好数据库表;而当我们的应用进行升级需要修改数据库表的结构时,这个时候就需要 对数据库表进行更新了;对于这两个操作,安卓给我们提供了**SQLiteOpenHelper**的两个方法, **onCreate**( )与**onUpgrade**( )来实现

方法解析：

> - **onCreate(database)**:首次使用软件时生成数据库表
> - **onUpgrade(database,oldVersion,newVersion)**:在数据库的版本发生变化时会被调用， 一般在软件升级时才需改变版本号，而数据库的版本是由程序员控制的，假设数据库现在的 版本是1，由于业务的变更，修改了数据库表结构，这时候就需要升级软件，升级软件时希望 更新用户手机里的数据库表结构，为了实现这一目的，可以把原来的数据库版本设置为2 或者其他与旧版本号不同的数字即可！

代码示例：

```
public class MyDBOpenHelper extends SQLiteOpenHelper {
    public MyDBOpenHelper(Context context, String name, CursorFacto
            int version) {super(context, "my.db", null, 1); }
    @Override
    //数据库第一次创建时被调用
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE person(personid INTEGER PRIMARY KE

    }
    //软件版本号发生改变时调用
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int ne
        db.execSQL("ALTER TABLE person ADD phone VARCHAR(12) NULL")
    }
}
```

代码解析：

> 上述代码第一次启动应用，我们会创建这个my.db的文件，并且会执行onCreate()里的方法， 创建一个Person的表，他又两个字段，主键personId和name字段；接着如我我们修改db的版本 号，那么下次启动就会调用onUpgrade()里的方法，往表中再插入一个字段！另外这里是插入 一个字段，所以数据不会丢失，如果是重建表的话，表中的数据会全部丢失，下一节我们会 来教大家如何解决这个问题！

流程小结：

> - **Step 1**：自定义一个类继承SQLiteOpenHelper类
> - **Step 2**：在该类的构造方法的super中设置好要创建的数据库名,版本号
> - **Step 3**：重写onCreate( )方法创建表结构
> - **Step 4**：重写onUpgrade( )方法定义版本号发生改变后执行的操作

3.如何查看我们生成的db文件

当我们调用上面的MyDBOpenhelper的对象的getWritableDatabase()就会在下述目录下创建我们的db 数据库文件：

Name	Size	Date	Time	Permissions
com.jay.sqlitedemo1		2015-09-02	12:24	drwxr-x--x
cache		2015-09-02	12:23	drwxrwx--x
databases		2015-09-02	12:24	drwxrwx--x
my.db	20480	2015-09-02	12:24	-rw-rw----
my.db-journal	8720	2015-09-02	12:24	-rw-------
lib		2015-09-02	12:24	lrwxrwxrwx

我们发现数据库有两个，前者是我们创建的数据库，而后者则是为了能让数据库支持事务而产生的 临时的日志文件！一般的大小是0字节！ 而在**File Explorer**里我们确是打不开文件的，连txt都打不开，何况是.db！ 所以下面给大家两条路选：

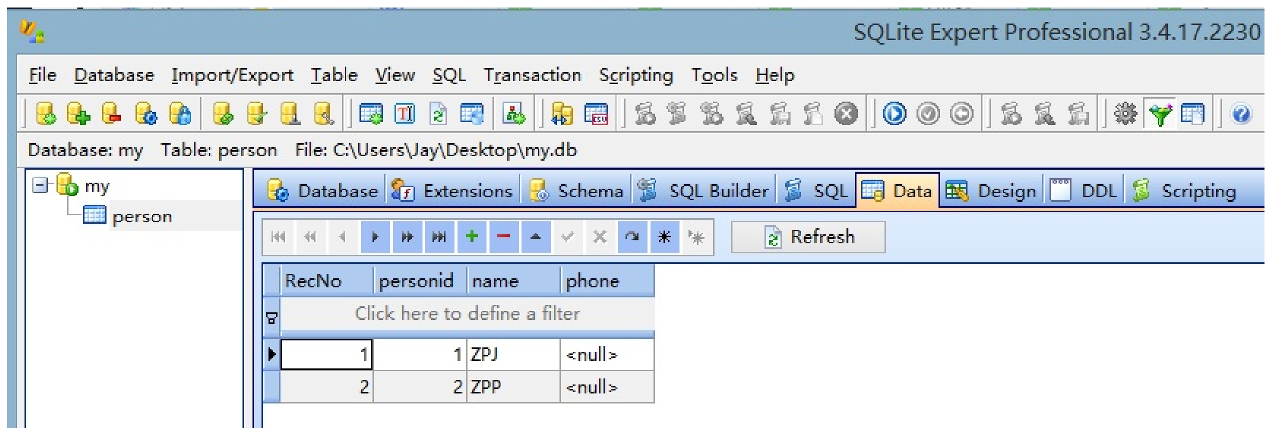
> - 1.先导出来，然后用SQLite的图形化工具查看
> - 2.配置adb环境变量后，通过adb shell来查看(命令行，装比利器)！

嗯，接着给大家演示上述两种方法，选自己喜欢的一种就可以了~~

方法1：使用SQLite图形化工具查看db文件

这类软件有很多，笔者用的是SQLite Expert Professional，当然你也可以使用其他工具 又需要的可以下载：SQLiteExpert.zip

把我们的db文件导出到电脑桌面，打开SQLiteExpert，界面如下：



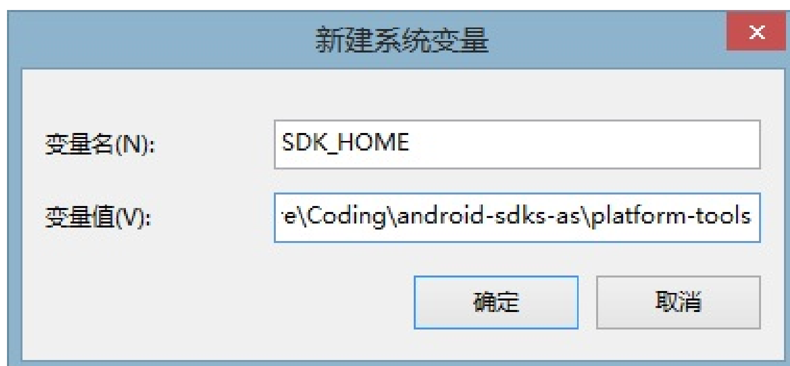
别问我怎么玩，导入db后自己慢慢玩，用法很简单，不懂百度~

至于方法二，本来是想试试的，后来发现sqlite命令找不到，试了几次就算了，后面用到在细扣，有兴趣可以找下郭霖的《第一行代码——Android》按着流程图试试！ 这里只贴前面的一部分，命令部分自己看书！

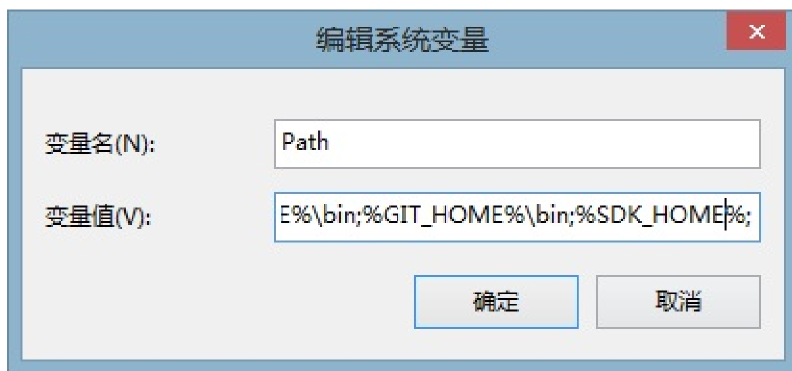
方法2：adb shell命令行带你装逼带你飞

1.配置SDK环境变量：

右键我的电脑 ——> 高级系统设置 -> 环境变量 -> 新建系统变量 -> 把SDK的platform-tools路径拷贝下： 比如笔者的：**C:\Software\Coding\android-sdks-as\platform-tools**

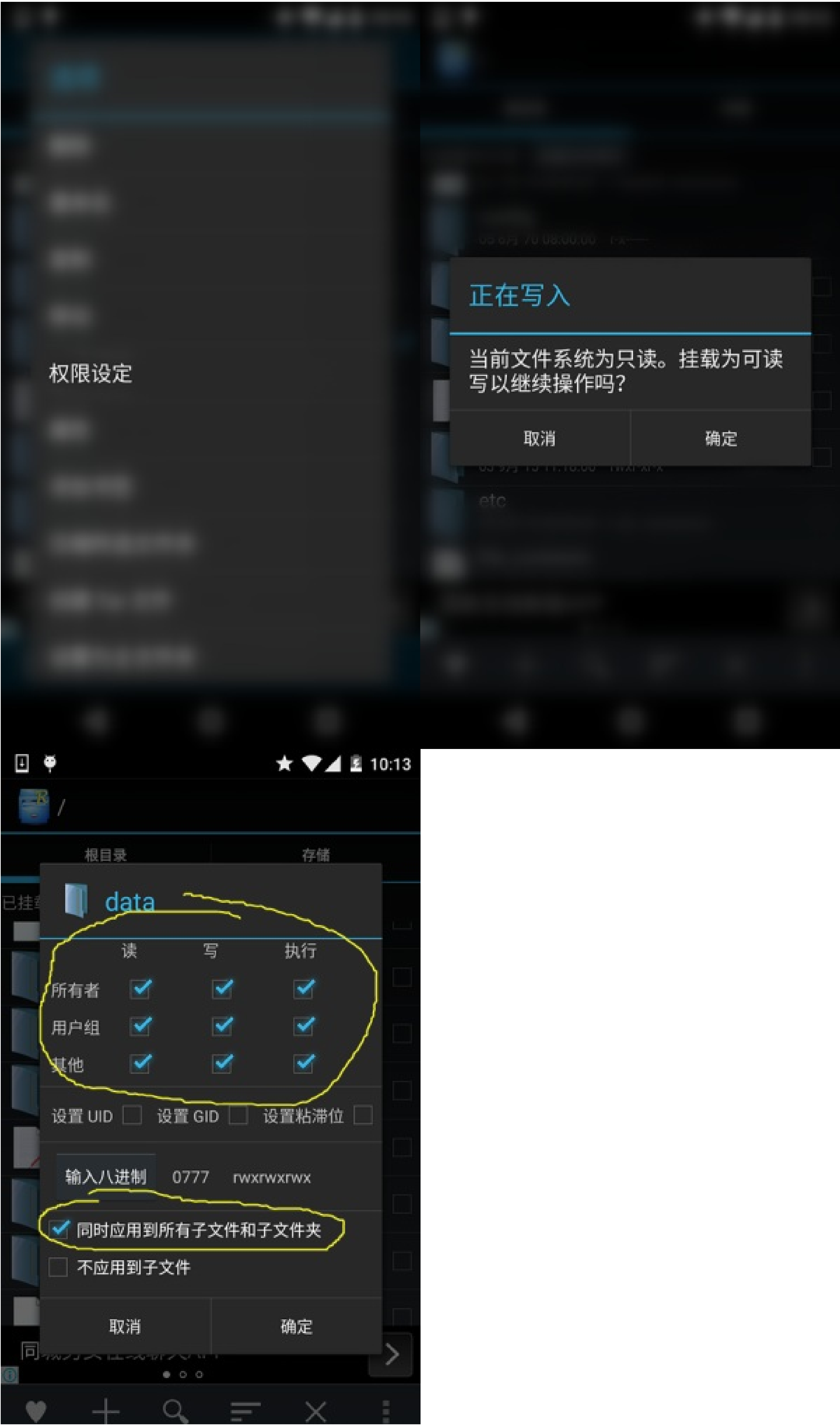


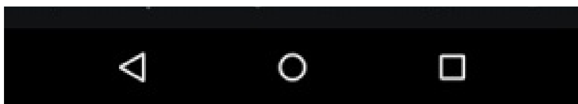
确定，然后再找到Path的环境变量，编辑，然后在结尾加上：**%SDK_HOME%;**



然后打开命令行，输入adb，唰唰唰一堆东西，就说明配置成功了！

————————重点———————— ：在执行后续命令行指令之前，针对你的测试的机器可能有几种： 1.原生模拟器：那行，你跳过这里，继续往下 2.**Genymotion**模拟器：没戏，Genymotion Shell执行不了下述命令 3.真机(已**root**)：那么你打开**File Explorer**看看data/data/目录下有东西没？没么？ 下面提供一个方法，就是先装个**RE**文件管理器，然后授予RE **Root**权限，接着来到根目录： 然后长按data目录，会弹出这样的对话框：


权限设定
正在写入
当前文件系统为只读。挂载为可读写以继续操作吗?
取消
确定
10:13
根目录
存储
data
读
写
执行
所有者
用户组
其他
设置 UID
设置 GID
设置粘滞位
输入八进制
0777
rwxrwxrwx
同时应用到所有子文件和子文件夹
不应用到子文件
取消
确定



接着等他慢慢修改权限，修改完毕后，我们再次打开DDMS的**File Explorer**，我们可以看到：

Name	Size	Date	Time	Permissions
com.jay.sqlitedemo1		2015-09-04	10:07	drwxrwxrwx
cache		2015-09-04	10:07	drwxrwxrwx
databases		2015-09-04	10:07	drwxrwxrwx
my.db	20480	2015-09-04	10:07	-rwxrwxrwx
my.db-journal	8720	2015-09-04	10:07	-rwxrwxrwx
lib		2015-09-04	10:07	lrwxrwxrwx

好的，可以看到data/data里的东西了！

2.进入**adb shell**，接着键入下述指令，来到我们**app**的**databases**目录下：

```
C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\Jay>adb shell
shell@hammerhead:/ $ cd /data/data/com.jay.sqlitedemo1/databases
cd /data/data/com.jay.sqlitedemo1/databases
shell@hammerhead:/data/data/com.jay.sqlitedemo1/databases $ ls
ls
my.db
my.db-journal
shell@hammerhead:/data/data/com.jay.sqlitedemo1/databases $ _
```

接着依次输入下述指令：

> - **sqlite3** my.db ：打开数据库文件
> - **.table** 查看数据库中有哪些表 接着你直接输入数据库语句就可以了，比如查询：Select * from person
> - **.schema**：查看建表语句
> - **.quit**：退出数据库的编辑
> - **.exit**：退出设备控制台

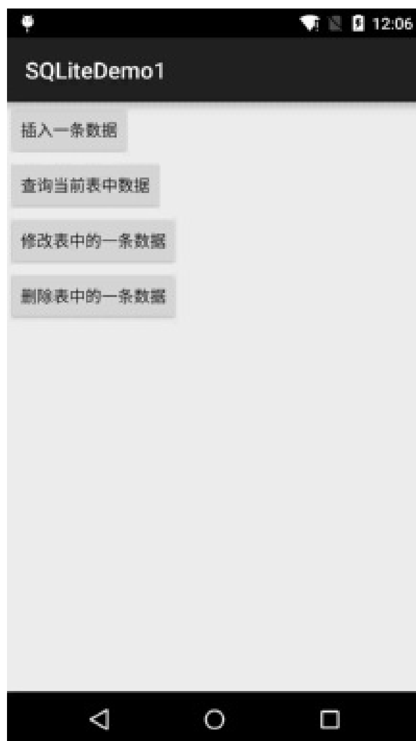
...因为system/bin/sh sqlite3: not found，这个问题，后面Sqlite命令的都用不了，要看效果图就自行查询郭大侠的书吧~而下面我们还是先导出db文件，然后用图形化的 数据库工具来查看！

4.使用Android提供的API操作SQLite

> 假如你没学过数据库相关的语法，或者你懒，不想写数据库语法，就可以使用Android给我们 提供的操作数据库的一些API方法，下面我们写个简单的例子来掩饰下这些API的用法！

代码示例：

运行效果图：



实现代码：

布局过于简单，就四个Button，就不贴了，直接贴**MainActivity.java**的代码：

```
public class MainActivity extends AppCompatActivity implements View

    private Context mContext;
    private Button btn_insert;
    private Button btn_query;
    private Button btn_update;
    private Button btn_delete;
    private SQLiteDatabase db;
    private MyDBOpenHelper myDBHelper;
    private StringBuilder sb;
    private int i = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
```

```
        myDBHelper = new MyDBOpenHelper(mContext, "my.db", null, 1)
        bindViews();
    }

    private void bindViews() {
        btn_insert = (Button) findViewById(R.id.btn_insert);
        btn_query = (Button) findViewById(R.id.btn_query);
        btn_update = (Button) findViewById(R.id.btn_update);
        btn_delete = (Button) findViewById(R.id.btn_delete);

        btn_query.setOnClickListener(this);
        btn_insert.setOnClickListener(this);
        btn_update.setOnClickListener(this);
        btn_delete.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        db = myDBHelper.getWritableDatabase();
        switch (v.getId()) {
            case R.id.btn_insert:
                ContentValues values1 = new ContentValues();
                values1.put("name", "呵呵~" + i);
                i++;
                //参数依次是：表名，强行插入null值得数据列的列名，一行记录的
                db.insert("person", null, values1);
                Toast.makeText(mContext, "插入完毕~", Toast.LENGTH_S
                break;
            case R.id.btn_query:
                sb = new StringBuilder();
                //参数依次是:表名，列名，where约束条件，where中占位符提供具
                //指定查询结果的排序方式
                Cursor cursor = db.query("person", null, null, null
                if (cursor.moveToFirst()) {
                    do {
                        int pid = cursor.getInt(cursor.getColumnInd
                        String name = cursor.getString(cursor.getCo
                        sb.append("id：" + pid + "：" + name + "\n"
                    } while (cursor.moveToNext());
                }
                cursor.close();
                Toast.makeText(mContext, sb.toString(), Toast.LENGT
                break;
            case R.id.btn_update:
                ContentValues values2 = new ContentValues();
                values2.put("name", "嘻嘻~");
                //参数依次是表名，修改后的值，where条件，以及约束，如果不指
                db.update("person", values2, "name = ?", new String
                break;
            case R.id.btn_delete:
                //参数依次是表名，以及where条件与约束
                db.delete("person", "personid = ?", new String[]{"3
                break;
```

```
        }
    }
}
```

5.使用SQL语句操作数据库

> 当然，你可能已经学过SQL，会写相关的SQL语句，而且不想用Android提供的这些API， 你可以直接使用SQLiteDatabase给我们提供的相关方法：
>
> - **execSQL**(SQL,Object[]):使用带占位符的SQL语句,这个是执行修改数据库内容的sql语句用的
> - **rawQuery**(SQL,Object[]):使用带占位符的SQL查询操作 另外前面忘了介绍下Curosr这个东西以及相关属性，这里补充下：——**Cursor**对象有点类似于JDBC中的ResultSet,结果集!使用差不多,提供一下方法移动查询结果的记录指针:
> - **move**(offset):指定向上或者向下移动的行数,整数表示向下移动;负数表示向上移动！
> - **moveToFirst**():指针移动到第一行,成功返回true,也说明有数据
> - **moveToLast**():指针移动到最后一样,成功返回true;
> - **moveToNext**():指针移动到下一行,成功返回true,表明还有元素！
> - **moveToPrevious**():移动到上一条记录
> - **getCount**( )获得总得数据条数
> - **isFirst**():是否为第一条记录
> - **isLast**():是否为最后一项
> - **moveToPosition**(int):移动到指定行

使用代码示例：

1.插入数据：

```
public void save(Person p)
{
    SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
    db.execSQL("INSERT INTO person(name,phone) values(?,?)",
                new String[]{p.getName(),p.getPhone()});
}
```

2.删除数据：

```
public void delete(Integer id)
{
    SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
    db.execSQL("DELETE FROM person WHERE personid = ?",
                new String[]{id});
}
```

3.修改数据：

```
public void update(Person p)
{
    SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
    db.execSQL("UPDATE person SET name = ?,phone = ? WHERE personic
        new String[]{p.getName(),p.getPhone(),p.getId()});
}
```

4.查询数据：

```
public Person find(Integer id)
{
    SQLiteDatabase db = dbOpenHelper.getReadableDatabase();
    Cursor cursor =  db.rawQuery("SELECT * FROM person WHERE persor
            new String[]{id.toString()});
    //存在数据才返回true
    if(cursor.moveToFirst())
    {
        int personid = cursor.getInt(cursor.getColumnIndex("personi
        String name = cursor.getString(cursor.getColumnIndex("name'
        String phone = cursor.getString(cursor.getColumnIndex("phor
        return new Person(personid,name,phone);
    }
    cursor.close();
    return null;
}
```

5.数据分页：

```
public List<Person> getScrollData(int offset,int maxResult)
{
    List<Person> person = new ArrayList<Person>();
    SQLiteDatabase db = dbOpenHelper.getReadableDatabase();
    Cursor cursor =  db.rawQuery("SELECT * FROM person ORDER BY per
        new String[]{String.valueOf(offset),String.valueOf(maxResul
    while(cursor.moveToNext())
    {
        int personid = cursor.getInt(cursor.getColumnIndex("personi
        String name = cursor.getString(cursor.getColumnIndex("name"
        String phone = cursor.getString(cursor.getColumnIndex("phon
        person.add(new Person(personid,name,phone)) ;
    }
    cursor.close();
    return person;
}
```

6.查询记录数：

```
public long getCount()
{
    SQLiteDatabase db = dbOpenHelper.getReadableDatabase();
    Cursor cursor =  db.rawQuery("SELECT COUNT (*) FROM person",nul
    cursor.moveToFirst();
    long result = cursor.getLong(0);
    cursor.close();
    return result;
}
```

PS：除了上面获取条数的方法外还可以使用cursor.getCount()方法获得数据的条数,但是SQL语句要改改！比如**SELECT * FROM person;**

本节小结：

> 本节给大家介绍了Android内置SQLite的基本用法，还是比较简单的，下一节再来研究点稍微 高级一点的东西，SQLite事务，应用更新数据库里数据怎么处理，以及数据库存储大二进制文件 的方法！好的，本节就到这里~

6.3.2 数据存储与访问之——又见SQLite数据库

本节引言：

> 学习完上一节，关于Android中的SQLite的基本操作，你就已经掌握了，而在本节我们将会学习一些稍微高级一点的东西，数据库事务，怎么将大二进制数据存储到数据库中，以及版本升级时 数据库如何处理！好的，开始本节内容！

1.SQLite事务

SQLite事务

什么是事务?

我们都知道数据库是面向多个用户的,如果每个时刻只有一名用户在操作数据库,其他用户需要等待,
这很影响数据库资源的使用！而多个人并发访问又容易发生问题,比如A用户查询某种表时并没有
找到X项,而在该用户还没完成操作之前;另一个用户插入了X项而A依旧认为没有X项！
举个形象的例子,A给B转100块,那么A的存储要减100,B的存款也增加100,两个条件要同时满足才能完成
完成交易(提交事务),但是假如B账户的钱没有增加,但是A已经扣了100块,这样显然是不可以的,
所以要把A的100块返还给他,就是恢复原样(事务回滚)

**总结来说就是两个或多个操作捆绑到一起的操作,只有所有操作都执行了,事务才算执行完毕,才提交
如果有一个操作没有执行的话,那么事务就会回滚,就是恢复原形,之前做的操作都会撤销！**

简单例子:　用户 1给用户2转账10元:

```
public void transaction()
{
	SQliteDatabase db = dbOpenHelper.getWritableDatabase();
	db.beginTransaction();	//开启事务
	try{
		db.execSQL("update person set amount = amount -10 where personid = 1");
		db.execSQL("update person set amount = amount +10 where personid = 1");
		db.setTransactionSuccessful();  //设置事务标志为true,表示提交事务
	}finally
	{
		db.endTransaction(); //结束事务
	}
}
```

相关方法介绍：

beginTransaction():开启事务
endTransaction():结束事务
结束事务的结果有两种,事务回滚或者提交,而决定回滚还是提交决定与一个标记,默认为false,即回滚;
所以想提交的话需要调setTransactionSuccessful()设置为true!
而这里使用finally块是为了避免某个语句执行发生错误,导致程序的意外停止,
后面结束事务的操作没执行！

简单点说就是：写在事务里的所有数据库操作都成功，事务提交，否则，事务回滚，就是回到前面 的状态——未执行数据库操作的时候！另外，前面我们也将了，在data/data/<包名>/database/目录 下除了有我们创建的db文件外，还有一个xxx.db-journal这个文件就是用来让数据库支持事务而 产生的 临时的日志文件！

2.SQLite存储大二进制文件

当然，一般我们很少往数据库中存储大二进制文件，比如图片，音频，视频等，对于这些我们一般 是存储文件路径，但总会有些奇葩的需求，某天你突然想把这些文件存到数据库里，下面我们以 图片为例子，将图片保存到SQLite中，以及读取SQLite中的图片！

保存图片到SQLite中,读取SQLite中的图片

1.保存图片到SQLite中:

①在创建数据库表的时候,需要创建一个BLOB的字段,用于存储二进制的值

```
db.execSQL("Create table test ( _id INTEGER PRIMARY KEY AUTOINCREMENT,head_img BLOB )");
```

②将图片转换为BLOB格式(这里是以ImageView为例的,如果是普通图片只需要转换成Bitmap再调用即可)

```
SQLiteDatabase db = mDBService.getWritableDatabase(); // 得到数据库
try {
        ByteArrayOutputStream outs = new ByteArrayOutputStream();
        ((BitmapDrawable) imageview.getDrawable()).getBitmap().compress(
                CompressFormat.PNG, 100, outs);//压缩为PNG格式,100表示跟原图大小一样
        Object[] args = new Object[] {outs.toByteArray() };
        db.execSQL("INSERT INTO test(head_img) values(?)", args);
        outs.close();
        db.close();
} catch (Exception e) {e.printStackTrace();}
```

2.读取SQLite中的图片:

```
SQLiteDatabase db = mDBService.getreadableDatabase();
Cursor cursor = db.rawQuery("SELECT head_img FROM test",null);
if(cursor != null)
{
        if(cursor.moveToFirst())
        {
        //取出图片保存到字节数组中
                byte[] img = cursor.getBlob(cursor.getColumnIndex("head_img"));
        }
}
if(cursor != null)cursor.close();
//将图片显示到ImageView上
if(img != null)
{
        ByteArrayInputStream bin = new ByteArrayInputStream(img);
        imageview.setImageDrawable(Drawable.createFromStream(bin,"img"));
}
```

3.SimpleCursorAdapter绑定数据库数据

当然，这个玩玩可以，还是不建议使用，尽管用起来很简单！其实在讲ContentProvider我们就使用过这个东西来绑定联系人列表！这里就不写实例了，直接上核心代码！需要的自己捣鼓捣鼓就好了，另外，现在我们一般很少自己写数据库的东西，一般是通过第三方的框架：ormlite，greenDao等，在进阶部分，我们会再来学习~

核心代码:

```
list = (ListView) findViewById(R.id.mylist);
MyDBOpenHelper mydb = new MyDBOpenHelper(this);
//查询数据
Cursor cursor = mydb.query("select personid AS _id,name from person", null);
//绑定数据
SimpleCursorAdapter simpleCursorAdapter = new SimpleCursorAdapter(this,
R.layout.listitem, cursor,
new String[]{"name"}, new int[]{R.id.listtext});
list.setAdapter(simpleCursorAdapter);
```

代码解析:

SimpleCursorAdapter()的参数依次为:this,Listview对应的列表项的布局id,cursor对象,对应字段,显示数据的组件
ps:可以显示多个数据哦！一一对应即可,另外,使用SimpleCursorAdapter需要保证数据库表中有名为_id的字段
不然是会报错的！

4.数据库升级的一些集锦

PS:好吧，这一块我并没有做过，始终是项目经验不够，公司的产品都是定位类的，刚看过公司项目，发现前人留下的代码是：onCreate()创建DB，然后onUpgrade()把前面的DB删掉，然后再调用onCreate()方法！看了几个版本的代码，发现并没有数据库升级的操作...没得借鉴，只能参考下别人的做法了，下面是小猪查阅资料后的一些归纳，如果有什么不对，欢迎指出，可能有些第三方的框架已经弄好了这个，时间关系，就不慢慢去考究了！知道可以留言，谢谢！

1）什么是数据库版本升级？怎么升级法？

> 答：假如我们开发了一款APP，里面用到了数据库，我们假定这个数据库版本为v1.0，在这个版本，我们创建了一个x.db的数据库文件，我们通过onCreate()方法创建了第一个table，t_user，里面有两个字段：_id,user_id；后面我们想增加一个字段user_name，这个时候我们就需要对数据库表的结构进行修改了，而我们可以把更新数据库的操作梵高onUpgrade() 方法中，我们只需要在实例化自定义SQLiteOpenHelper的时候，修改版本号，比如把1改成2这样，就会自动调用onUpgrade()的方法了！另外，对于每个数据库版本我们都应该做好相应的记录(文档)，类似于下面这种:

数据库版本	andoid对应版本	内容
v1.0	1	第一个版本，包含两个字段...
v1.1	2	数据保留，新增user_name字段

2）一些疑问以及相关解决方案

①应用升级，数据库文件是否会删除？

> 答：不会！数据什么的都在！

②如果我想删除表中某个字段或者增加一个新的字段，原先的数据还在吗？

> 答：在的！

③你刚说的那种粗暴的更新数据库版本的方式，不保留数据的，可以贴下吗？

> 答：可以，这里用的是第三方的ormlite，你也可以自己写数据库创建以及删除的代码：

```
@Override
public void onCreate(SQLiteDatabase db, ConnectionSource connectionSource) {
    try {
        TableUtils.createTable(connectionSource, UserEntity.class);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Override
public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource,
        int arg2, int arg3) {
    try {
        TableUtils.dropTable(connectionSource, UserEntity.class, true);
        onCreate(db, connectionSource);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

④比如是这种，假如我们已经升级到第三个版本了，我们在第二个版本增加了一个表，然后第三个版本也增加了一个表，加入用户直接从第一个版本升级到第三个版本，这样没经过第二个版本，就没有增加的那个表，这可怎么破？

> 答：很简单，我们可以在onUpgrade()里写一个switch(),结构如下：
>
> ```
> public void onUpgrade(SQLiteDatabase db, ConnectionSource conne
>             int arg2, int arg3) {
>     switch(arg2){
>         case 1:
>             db.execSQL(第一个版本的建表语句);
>         case 2:
>             db.execSQL(第二个版本的建表语句);
>         case 3:
>             db.execSQL(第三个版本的建表语句);
>     }
> }
> ```
>
> 细心的你可能发现这里并没有写break，这就对了，这是为了保证跨版本升级时，每次数据库修改都能全部执行到！这样可以保证表结构都是最新的！另外不一定是建表语句，修改表结构也可以哦！

⑤旧表的设计太糟糕，很多字段要改，改动太多，想建一个新表，但是表名要一样而且以前的一些数据要保存到新表中！

> 答：呵呵，给你跪了，当然，也有解决办法，下面说下思路：
>
> 1.将旧表改名成临时表: **ALTER TABLE User RENAME TO _temp_User;**
>
> 2.创建新表: **CREATE TABLE User (u_id INTEGER PRIMARY KEY,u_name VARCHAR(20),u_age VARCHAR(4));**
>
> 3.导入数据； **INSERT INTO User SELECT u_id,u_name,"18" FROM _temp_User;** //原表中没有的要自己设个默认值
>
> 4.删除临时表； **DROP TABLE_temp_User;**

本节小结：

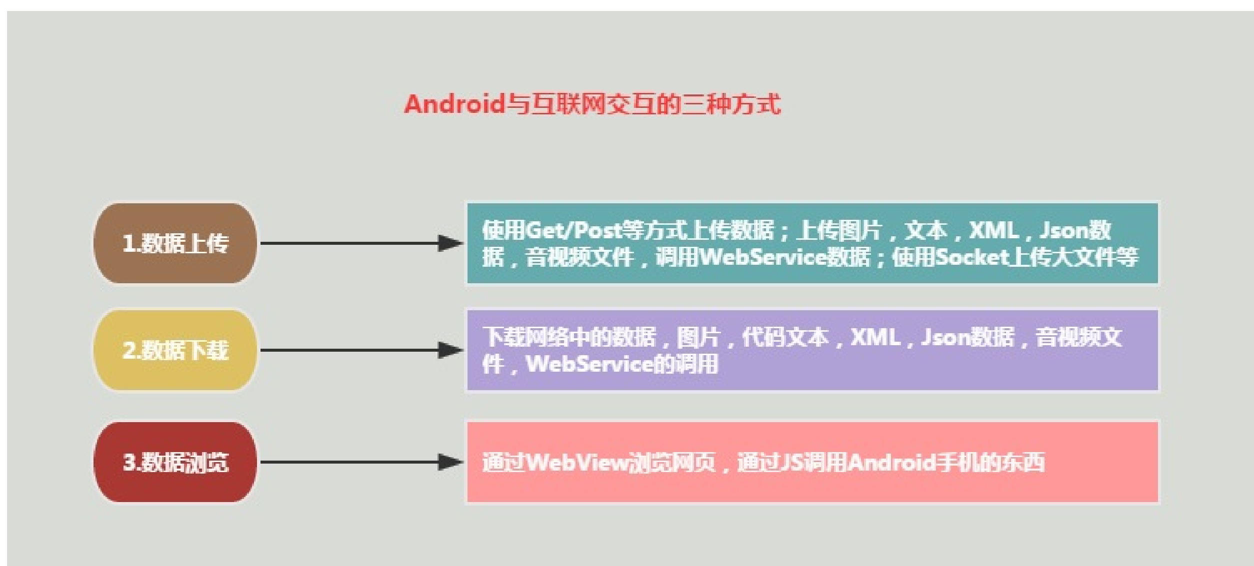
> 好的，本节我们对SQLite的事务，大二进制存储，SimpleCursorAdapter以及数据库升级的 一些问题进行了探究，而关于SQLite的东西，我们暂时就学这么多，关于第三方的使用，以及 一些高深的话题，我们到进阶的时候再和大家一起去研究~本节就到这里，谢谢~

7.1.1 Android网络编程要学的东西与Http协议学习

本节引言：

> 不知不觉终于来到Android网络编程这一章节，前面我们玩的都是单机，肯定是不过瘾是吧，本节开始我们来学习Android网络编程相关的一些东西：Android端网络编程要干嘛？Http协议 的学习，使用自带扣脚Json解析类解析Json，XML解析的几种常用方式，HttpUrlConnection和 HttpClient的使用，文件的上传，下载；WebService的使用，WebView，Socket通信的使用等！
>
> 另外我们是客户端，服务端的内容不在我们的范畴，而且小猪也不擅长，我们的最低要求是：能够掌握获取与解析服务器反馈的数据的能力！好的，话不多说，开始本节内容！

1.Android与互联网交互的三种方式



2.初识Http协议

> 实际开发中我们和服务端打交道一般用得都是基于Http协议的通信，所以学好Http协议是非常 重要的，当然，我们不用过于考究一些细节的东西，有个大体的了解即可！都是一些概念性的东西！

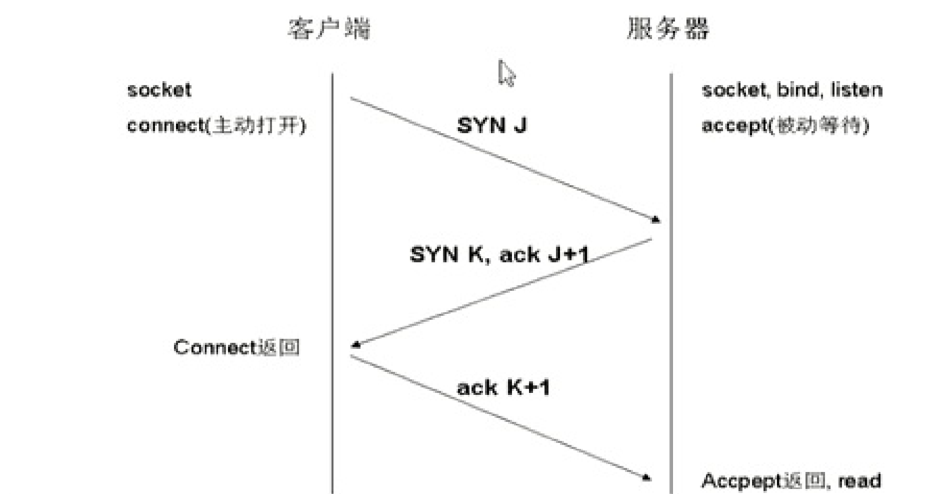
1）什么是Http协议？

> 答：hypertext transfer protocol（超文本传输协议），TCP/IP协议的一个应用层协议，用于 定义WEB浏览器与WEB服务器之间交换数据的过程。客户端连上web服务器后，若想获得web服务器 中的某个web资源，需遵守一定的通讯格式，HTTP协议用于定义客户端与web服务器通迅的格式。

2）Http 1.0 与 Http 1.1的区别

> 答：1.0协议，客户端与web服务器建立连接后，只能获得一个web资源！ 而1.1协议，允许客户端与web服务器建立连接后，在一个连接上获取多个web资源！

3）Http协议的底层工作流程：

> 答：我们先要知道两个名词：
>
> - **SYN**(synchronous):TCP/IP建立连接时使用的握手信号
> - **ACK**(Acknowledgement):确认字符，确认发来的数据已经接受无误
>
> 接着就到**TCP/IP**三次握手的概念：
>
> - 客户端发送syn包(syn = j)到服务器，进入SYN_SEND状态，然后等待服务器确认
> - 服务器收到syn包,确认客户的syn(ack = j + 1),同时在自己也发送一个SYN包(syn=k)， 即SYN + ACK包，服务器进入SYN_RECV状态
> - 客户端收到SYN + ACK包，向服务器发送确认包ACK(ack = k +1),发送完毕后，客户端与服务端 进入ESTABLISHED状态，完成三次握手，然后两者开始传送数据
>
> 如果还不是很清晰，我们再来看三次握手的示意图：
>
> 
>
>
> 了解了是吧，然后我们就来看看Http操作的一个流程了：
>
> - 用户点击浏览器上的url(超链接)，Web浏览器与Web服务器建立连接

- 建立连接后，客户端发送请求给服务器，请求的格式为：统一资源标识符(URL)+协议版本号(一般是1.1)+MIME信息(多个消息头)+一个空行
- 服务端收到请求后，给予相应的返回信息，返回格式为：协议版本号 + 状态行(处理结果) + 多个信息头 + 空行 + 实体内容(比如返回的HTML)
- 客户端接收服务端返回信息，通过浏览器显示出来，然后与服务端断开连接；当然如果中途 某步发生错误的话，错误信息会返回到客户端，并显示，比如：经典的404错误！

对于上面的流程如果还不清晰，我们可以使用HttpWatch或者firefox抓下包：PS:测试网站是小猪的学校的教务系统，输入账号密码后请求登陆，我们可以看到下述信息：

HTTP请求包含的内容:

849 字节 发送到 127.0.0.1:18186　　Find　Export

```
POST /default2.aspx HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg,
Referer: http://jiaowu.zhbit.com/
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; QQDownload 7
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: jiaowu.zhbit.com
Content-Length: 199
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: ASP.NET_SessionId=ugkdpubz0aw1m5uhoirrac55

__VIEWSTATE=dDwyODE2NTM0OTg7Oz5v%2BDe3E3S3%2BljsAqmagIDcdbdODw%3D%3D&txtUserName=110202051015&T
```

发送Http请求

HTTP响应包括的内容:

482 字节 接收于 127.0.0.1:61396

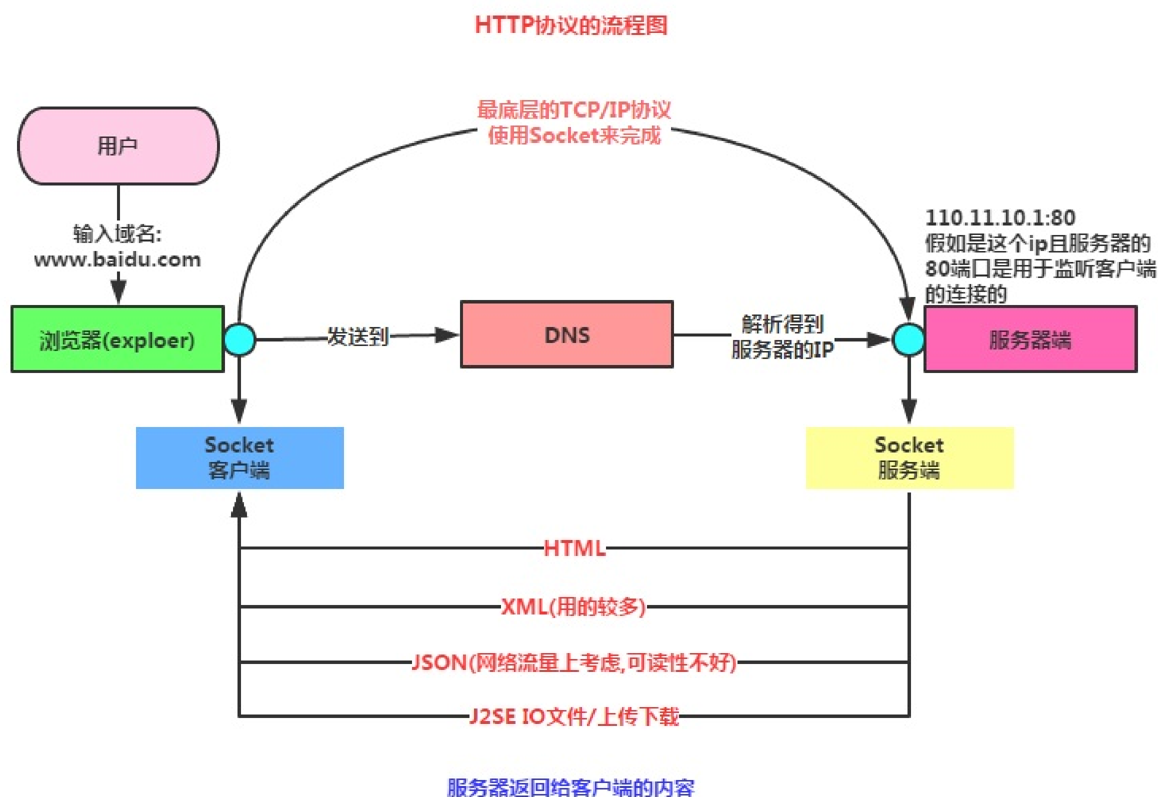
```
HTTP/1.1 302 Found
Cache-Control: no-cache, no-store
Pragma: no-cache,no-cache
Content-Length: 146
Content-Type: text/html; charset=gb2312
Expires: -1
Location: /xs_main.aspx?xh=110202051015
Server: Microsoft-IIS/7.0
X-AspNet-Version: 1.1.4322
P3P: CP=CAO PSA OUR
X-Powered-By: ASP.NET
Date: Mon, 01 Jun 2015 13:44:27 GMT

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href='/xs_main.aspx?xh=110202051015'>here</a>.</h2>
</body></html>
```

协议版本　各种头　Http回应的内容　空行　返回的内容

这就一目了然了是吧！

4）Http协议的业务流程



5）Http的几种请求方式

> 实际开发中我们用得较多的方式是Get和Post，但是实际开发可能还会用到其他请求方式，比如PUT，小猪的实际项目中就用到了，下面为了方便大家，就把所有的请求方式列出来吧：
>
> - **Get**：请求获取Request-URI所标识的资源
> - **POST**：在Request-URI所标识的资源后附加新的数据
> - **HEAD** 请求获取由Request-URI所标识的资源的响应信息报头
> - **PUT**：请求服务器存储一个资源，并用Request-URI作为其标识
> - **DELETE**：请求服务器删除Request-URI所标识的资源
> - **TRACE**：请求服务器回送收到的请求信息，主要用于测试或诊断
> - **CONNECT**：保留将来使用
> - **OPTIONS**：请求查询服务器的性能，或者查询与资源相关的选项

6）Get和Post的对比

> 用得最多的两个，当然要做下对比啦！
>
> - **GET**：在请求的URL地址后以?的形式带上交给服务器的数据，多个数据之间以&进行分隔， 但数据容量通常不能超过2K，比如:http://xxx?username=…&pawd=…这种就是GET
> - **POST**: 这个则可以在请求的实体内容中向服务器发送数据，传输没有数量限制
> - 另外要说一点，这两个玩意都是发送数据的，只是发送机制不一样，不要相信网上说的 "GET获得服务器数据，POST向服务器发送数据"!!另外GET安全性非常低，Post安全性较高， 但是执行效率却比Post方法好，一般查询的时候我们用GET，数据增删改的时候用POST！！

7）Http状态码合集

> 当然，这些状态码只是要给参考，实际上决定权在服务器端(后台的)手上，一种方案是请求后， 服务器返回给我们的是状态，或者另一种，在应用不用弄多语言版本的时候最好用，直接返回 一串结果信息的Json给我们，我们直接显示就好，这样可以偷懒不少！下面列下状态码合集，参考 下就好：
>
> - 100~199：成功接受请求，客户端需提交下一次请求才能完成整个处理过程
> - 200: OK，客户端请求成功
> - 300~399：请求资源已移到新的地址(302,307,304)
> - 401：请求未授权，改状态代码需与WWW-Authenticate报头域一起使用
> - 403：Forbidden，服务器收到请求，但是拒绝提供服务
> - 404：Not Found，请求资源不存在，这个就不用说啦
> - 500：Internal Server Error，服务器发生不可预期的错误
> - 503：Server Unavailable，服务器当前不能处理客户端请求，一段时间后可能恢复正常

8）Http协议的特点

> 概念性的东西，知道就好，别去背，百度百科的东西，直接复制粘贴：
>
> **1.** 支持客户/服务器模式。
>
> **2.** 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、 HEAD、POST。每种方法规定了客户与服务器联系的类型不同。 由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
>
> **3.** 灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
>
> **4.** 无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求， 并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
>
> **5.** 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。 缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每 次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

PS:关于OSI七层协议以及TCP四层模型就不在基础系列讲解了~有兴趣可以自行了解下！

本节小结：

> 本节讲解了我们在Android开发中涉及到网络方面的要完成的工具，以及讲解了Http协议的 相关概念，相信大家对Android移动端与服务器交互已经有了个模糊的映像，下节我们来研究 Http协议的请求头与响应头！本节就到这里，谢谢~

7.1.2 Android Http请求头与响应头的学习

本节引言：

> 上节中我们对Android涉及的网络编程进行了了解，也学习了下Http的基本概念，而本节我们 要学习的是Http的请求头与响应头，当然，可以把也可以把这节看作文档，用到的时候来查查 即可！

1.HTTP请求之消息头:

> 这里贴下上一节给出的图,根据下面给出的表，大家自己感受下相关请求头的作用吧: PS:第一行是请求行:请求方式 + 资源名称 + HTTP协议版本号，另外请求头只是给服务端的一个 信息而已或者说一个简单，至于怎么处理，还是由服务端来决定的！



HTTP Request Header请求头信息对照表:

Header	解释	示例
Accept	指定客户端能够接收的内容类型	Accept: text/plain, text/html
Accept-Charset	浏览器可以接受的字符编码集。	Accept-Charset: iso-8859-5
Accept-Encoding	指定浏览器可以支持的web服务器返回内容压缩编码类型。	Accept-Encoding: compress, gzip
Accept-Language	浏览器可接受的语言	Accept-Language: en,zh

Accept-Ranges	可以请求网页实体的一个或者多个子范围字段	Accept-Ranges: bytes
Authorization	HTTP授权的授权证书	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Cache-Control	指定请求和响应遵循的缓存机制	Cache-Control: no-cache
Connection	表示是否需要持久连接。（HTTP 1.1默认进行持久连接）	Connection: close
Cookie	HTTP请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。	Cookie: $Version=1; Skin=new;
Content-Length	请求的内容长度	Content-Length: 348
Content-Type	请求的与实体对应的MIME信息	Content-Type: application/x-www-form-urlencoded
Date	请求发送的日期和时间	Date: Tue, 15 Nov 2010 08:12:31 GMT
Expect	请求的特定的服务器行为	Expect: 100-continue
From	发出请求的用户的Email	From: user@email.com
Host	指定请求的服务器的域名和端口号	Host: www.zcmhi.com
If-Match	只有请求内容与实体相匹配才有效	If-Match: "737060cd8c284d8af7ad3082f209582d"
If-Modified-Since	如果请求的部分在指定时间之后被修改则请求成功，未被修改则返回304代码	If-Modified-Since: Sat, 29 Oct 2010 19:43:31 GMT
	如果内容未改变返回304代码，	

If-None-Match	参数为服务器先前发送的Etag，与服务器回应的Etag比较判断是否改变	If-None-Match: "737060cd8c284d8af7ad3082f209582d"
If-Range	如果实体未改变，服务器发送客户端丢失的部分，否则发送整个实体。参数也为Etag	If-Range: "737060cd8c284d8af7ad3082f209582d"
If-Unmodified-Since	只在实体在指定时间之后未被修改才请求成功	If-Unmodified-Since: Sat, 29 Oct 2010 19:43:31 GMT
Max-Forwards	限制信息通过代理和网关传送的时间	Max-Forwards: 10
Pragma	用来包含实现特定的指令	Pragma: no-cache
Proxy-Authorization	连接到代理的授权证书	Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Range	只请求实体的一部分，指定范围	Range: bytes=500-999
Referer	先前网页的地址，当前请求网页紧随其后,即来路	Referer: http://blog.csdn.net/coder_pig
TE	客户端愿意接受的传输编码，并通知服务器接受接受尾加头信息	TE: trailers,deflate;q=0.5
Upgrade	向服务器指定某种传输协议以便服务器进行转换（如果支持）	Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11
User-Agent	User-Agent的内容包含发出请求的用户信息	User-Agent: Mozilla/5.0 (Linux; X11)
Via	通知中间网关或代理服务器地址，通信协议	Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)

Warning	关于消息实体的警告信息	Warn: 199 Miscellaneous warning

2.HTTP响应之响应头:

> 同样给出上节的图: PS:第一行依次是:协议版本号 状态码 302表示这里没有，但是另外一个地方有，通过Location页面重定向了

```
482 字节 接收于 127.0.0.1:61396
HTTP/1.1 302 Found
Cache-Control: no-cache, no-store
Pragma: no-cache,no-cache
Content-Length: 146
Content-Type: text/html; charset=gb2312
Expires: -1
Location: /xs_main.aspx?xh=110202051015
Server: Microsoft-IIS/7.0
X-AspNet-Version: 1.1.4322
P3P: CP=CAO PSA OUR
X-Powered-By: ASP.NET
Date: Mon, 01 Jun 2015 13:44:27 GMT

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href='/xs_main.aspx?xh=110202051015'>here</a>.</h2>
</body></html>
```

协议版本 各种头 Http回应的内容 空行 返回的内容

HTTP Responses Header 响应头信息对照表：

Header	解释	示例
Accept-Ranges	表明服务器是否支持指定范围请求及哪种类型的分段请求	Accept-Ranges: bytes
Age	从原始服务器到代理缓存形成的估算时间（以秒计，非负）	Age: 12
Allow	对某网络资源的有效的请求行为，不允许则返回405	Allow: GET, HEAD
Cache-Control	告诉所有的缓存机制是否可以缓存及哪种类型	Cache-Control: no-cache
Content-	web服务器支持的返回内容压缩编	Content-Encoding: gzip

	码类型	
Content-Language	响应体的语言	Content-Language: en,zh
Content-Length	响应体的长度	Content-Length: 348
Content-Location	请求资源可替代的备用的另一地址	Content-Location: /index.htm
Content-MD5	返回资源的MD5校验值	Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==
Content-Range	在整个返回体中本部分的字节位置	Content-Range: bytes 21010-47021/47022
Content-Type	返回内容的MIME类型	Content-Type: text/html; charset=utf-8
Date	原始服务器消息发出的时间	Date: Tue, 15 Nov 2010 08:12:31 GMT
ETag	请求变量的实体标签的当前值	ETag: "737060cd8c284d8af7ad3082f209582d"
Expires	响应过期的日期和时间	Expires: Thu, 01 Dec 2010 16:00:00 GMT
Last-Modified	请求资源的最后修改时间	Last-Modified: Tue, 15 Nov 2010 12:45:26 GMT
Location	用来重定向接收方到非请求URL的位置来完成请求或标识新的资源	Location: http://blog.csdn.net/coder_pig
Pragma	包括实现特定的指令，它可应用到响应链上的任何接收方	Pragma: no-cache
Proxy-Authenticate	它指出认证方案和可应用到代理的该URL上的参数	Proxy-Authenticate: Basic

3.代码验证响应头的作用：

> 好了，看了那么多概念的东西，不动动手怎么行呢？是吧，那我们就写一些简单的代码来验证一些 常用的响应头的作用吧，以便加深我们的了解，这里的话服务端就用最简单的Servlet来实现，如果不会 Java Web的朋友只需将代码拷一拷，配置下web.xml，把Servlet的类名扣上，比如:

```
<servlet>  <servlet-name>FirstServlet</servlet-name>  <servlet-clas
```

改成对应的类名即可！

1）通过Location实现页面重定向

实现代码：

```
package com.jay.http.test;  import java.io.IOException;  import jav
```

运行结果：

当我们去访问:http://localhost:8080/HttpTest/ServletOne的时候，我们会发现页面跳转到了百度，接着我们用FireFox的开发者工具:可以看到我们发出的HTTP的内容:

响应头：

Content-Length: 0
Date: Tue, 02 Jun 2015 03:08:02 GMT
Location: http://www.baidu.com
Server: Apache-Coyote/1.1

2）通过Content-Encoding告诉浏览器数据的压缩格式

实现代码:

```
package com.jay.http.test;  import java.io.ByteArrayOutputStream;
```

运行结果：

控制台输出:

原始数据长度:478

浏览器输出:

Fresh air and sunshine can have
exercise is also a fantastic way
air, let the sun hit our face, g

再看看我们的HTTP内容:

响应头：

Content-Encoding: gzip
Content-Length: 285
Date: Tue, 02 Jun 2015 03:50:27 GMT
Server: Apache-Coyote/1.1

```
1  Fresh air and sunshine can have an amazi
```

这个**gzip**压缩字符串对于简单的字符串压缩，效率不高，比如小猪本来写的是一个一首静夜诗的字符串， 后来发现压缩过后的大小，竟然比原先的还要大=-=...

3）通过**content-type**，设置返回的数据类型

> 服务端返回的有时可能是一个text/html，有时也可能是一个image/jpeg，又或者是一段视频video/avi 浏览器可以根据这个对应的数据类型，然后以不同的方式将数据显示出来！好吧，这里我们弄一个读PDF的

实现代码：

```
package com.jay.http.test;  import java.io.IOException;  import jav
```

运行结果：

在浏览器上输入:http://localhost:8080/HttpTest/ServletThree

好哒，果然可以读到PDF了,对了，这个PDF我已经丢在webroot的file目录下，不然会报空指针哦~:

当然，你也可以试着去播放一段音乐或者视频，只需修改下content-type这个参数而已

下面顺便给出个HTTP Content-type的对照表吧: HTTP Content-type的对照表



Android 编码规范

4）通过refresh响应头，让浏览器隔几秒后跳转至别的页面

恩呢，一般我们可能有这样的需求，比如每隔几秒刷新一次页面，又或者加载某个页面几秒后 又跳转至另一个页面，那么refresh可以满足你的需要~

实现代码：

```
package com.jay.http.test;  import java.io.IOException;  import jav
```

运行结果：

> - 1的话每隔2秒刷新一次页面，我们可以看到显示的数字是递增的，另外doGet方法也一直被调用，说明页面真的是刷新的！
> - 2的话进入页面后5s，就自己跳转到百度了~

5）通过content-dispostion响应头，让浏览器下载文件

> 这个很简单，我们只需把③中设置Content-type的一行去掉，然后加上：**resp.setHeader("content-disposition", "attachment;filename=Android.pdf");**

实现代码：

```
package com.jay.http.test;  import java.io.IOException;  import jav
```

运行结果：



本节小结：

> 本节给大家介绍了Http中的请求头和响应头，也写了几个关于响应头调浏览器的一些示例， 相信经过本章，大家对于Http协议更加了解了，下节我们来学习Android给我们提供的Http 请求方式:HttpURLConnection！好的，本节就到这里，谢谢~ 对了，本节demo下载：下载 HttpTest.zip

7.1.3 Android HTTP请求方式:HttpURLConnection

本节引言：

> 前面两节我们学习的都是一些概念性的东西，Http的协议以及协议头的一些东东，而本节我们 就要堆码了，而本节学习的是Android为我们提供的Http请求方式之一：HttpURLConnection， 除了这种，还有一种还有一种HttpClient，后者我们会下一节讲！不过前者一旦请求复杂起来，使用起来 非常麻烦，而后者我们Java抓包也经常会用到，是Apache的，毕竟不是谷歌亲儿子，而在4.4版本 HttpURLConnection已被替换成OkHttp了！好吧，与时俱进，决定讲完HttpClient也来会会这个 OkHttp！对了，一般我们实际开发并不会用HttpURLConnection和HttpClient，使用别人封装 好的第三方网络请求框架，诸如：Volley，android-async-http，loopj等，因为网络操作涉及到 异步以及多线程，自己动手撸的话，很麻烦，所以实际开发还是直接用第三方！！当然学习下也 无妨，毕竟第三方也是在这些基础上撸起来的，架构逼格高，各种优化！好的，话不多说，开始 本节内容！

1.HttpURLConnection的介绍

> 答：一种多用途、轻量极的HTTP客户端，使用它来进行HTTP操作可以适用于大多数的应用程序。 虽然HttpURLConnection的API提供的比较简单，但是同时这也使得我们可以更加容易地去使 用和扩展它。继承至URLConnection，抽象类，无法直接实例化对象。通过调用**openCollection**() 方法获得对象实例，默认是带gzip压缩的；

2.HttpURLConnection的使用步骤

使用HttpURLConnection的步骤如下：

> - 创建一个URL对象： **URL url = new URL(http://www.baidu.com);**
> - 调用URL对象的openConnection( )来获取HttpURLConnection对象实例：**HttpURLConnection conn = (HttpURLConnection) url.openConnection();**
> - 设置HTTP请求使用的方法:GET或者POST，或者其他请求方式比如：PUT **conn.setRequestMethod("GET");**
> - 设置连接超时，读取超时的毫秒数，以及服务器希望得到的一些消息头 **conn.setConnectTimeout(6*1000); conn.setReadTimeout(6 * 1000);**
> - 调用getInputStream()方法获得服务器返回的输入流，然后输入流进行读取了 **InputStream in = conn.getInputStream();**
> - 最后调用disconnect()方法将HTTP连接关掉 **conn.disconnect();**

PS:除了上面这些外,有时我们还可能需要对响应码进行判断,比如200: if(conn.getResponseCode() != 200)然后一些处理 还有，可能有时我们 并不需要传递什么参数，而是直接去访问一个页面，我们可以直接用： final InputStream in = new URL("url").openStream(); 然后直接读流，不过这个方法适合于直接访问页面的情况，底层实现其实也是 return openConnection().getInputStream()，而且我们还不能设置一些 请求头的东 东，所以要不要这样写，你自己要掂量掂量！

3.HttpURLConnection使用示例

> 这里我们主要针对GET和POST请求写两个不同的使用示例，我们可以conn.getInputStream() 获取到的是一个流，所以我们需要写一个类将流转化为二进制数组！工具类如下：

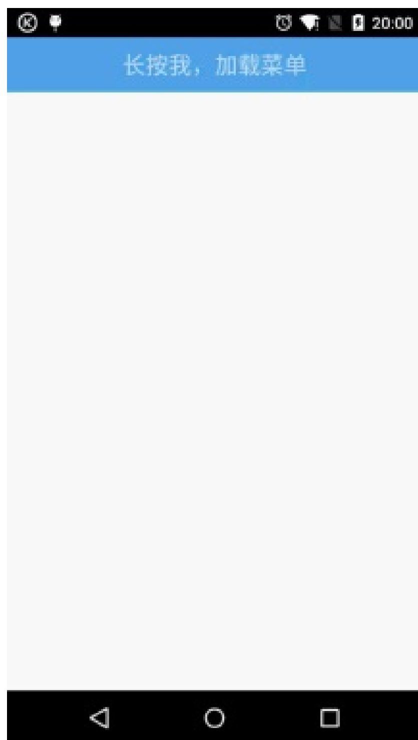
StreamTool.java:

```
/**
 * Created by Jay on 2015/9/7 0007.
 */ public class StreamTool { //从流中读取数据 public static
```

接下来就可以开始撸我们的示例了！

1）HttpURLConnection发送GET请求代码示例

运行效果图：



核心部分代码：

布局：**activity_main.xml**

```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/ar
```

获取数据类：**GetData.java**:

```
/**
 * Created by Jay on 2015/9/7 0007.
 */  public  class  GetData  {  // 定义一个获取网络图片数据的方法:  publ
```

MainActivity.java：

```
public  class  MainActivity  extends  AppCompatActivity  {  private
```

最后别忘了加上联网权限：

```
<uses-permission  android:name="android.permission.INTERNET"  />
```

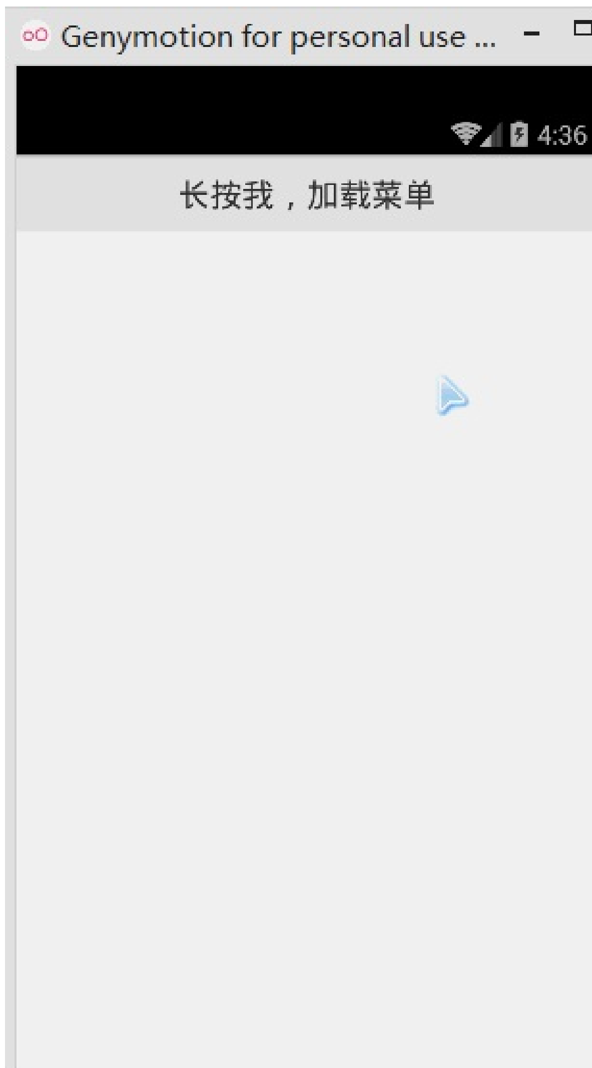
注意事项：

> 用handler的原因就不用讲了吧~ 另外我们加载html代码的使用的是webView的loadDataWithBaseURL而非LoadData， 如果用LoadData又要去纠结中文乱码的问题，so…用loadDataWithBaseURL就可以不用纠结那么多了 另外有些页面可能需要我们提交一些参数，比如账号密码:我们只需把对应参数拼接到url尾部即可，比如: http://192.168.191.1:8080/ComentServer/LoginServlet?passwd=123&name=Jack 然后服务端getParamater("passwd")这样就可以获得相应的参数了，我们请求时这些东西都会看得清清楚楚，所以说GET方式并不安全！另外还有一点要注意的就是Android从4.0开始就不允许在非UI线程中进行UI操作!

2）HttpURLConnection发送POST请求代码示例

> 有GET自然有POST，我们通过openConnection获取到的HttpURLConnection默认是进行Get请求的, 所以我们使用POST提交数据，应提前设置好相关的参数:conn.setRequestMethod("POST"); 还有:conn.setDoOutput(true);conn.setDoInput(true);设置允许输入，输出 还有:conn.setUseCaches(false); POST方法不能缓存，要手动设置为false, 具体实现看代码:

运行效果图:



核心代码：

PostUtils.java

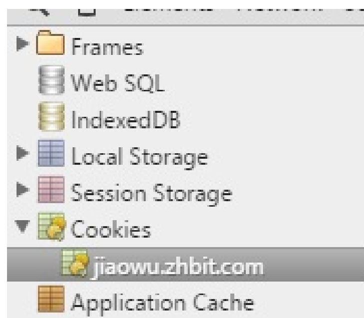
```
public  class  PostUtils  {  public  static  String LOGIN_URL =  "h
                                message.write(buffer,  0, len);  }  // 释放
```

PS：因为电脑没装MyEclipse，而且时间关系，就不另外写demo了，用回之前的Eclipse的那个demo！ 其实从直接看核心代码就够了~ 代码下载：HttpURLConnection例子.zip

4.Cookie问题的处理

> 说这个之前，首先我们要理解两个概念：**Session**和**Cookie** Cookie只是Session机制的一种常用形式，我们也可以使用其他方式来作为客户端的一个唯一标识， 这个由服务器决定，唯一能够证明一个客户端标识就好！除了这种方式外，我们还可以使用URL重写！ 方法来实现！所以以后别傻傻的跟别人说：Session不就是Cookie！

下面通过一个例子来帮助大家理解这个Cookie： 小猪输入账号密码后登陆下学校的教务系统，然后访问课表信息成功， 然后如果你用的是Chrome，按F12进入开发模式:来到Resources界面可以看到我们的Cookies:



Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure	First-Party
ASP.NET_SessionId	ul3f4snfnnrrpo55swv34yry	jiaowu.zhbit.com	/	Session	41			

点击后我们可以看到里面保存的内容，由名称；值；cookie所在的域(domain)； cookie所在的目录(path)Asp.net默认为/即根目录；过期时间；Cookie大小：

我们可以看到请求头中有一个Cookie的字段：

```
▼Request Headers    view source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
   Accept-Encoding: gzip, deflate, sdch
   Accept-Language: zh-CN,zh;q=0.8
   Cache-Control: max-age=0
   Connection: keep-alive
   Cookie: ASP.NET_SessionId=ul3f4snfnnrrpo55swv34yry
   Host: jiaowu.zhbit.com
   Referer: http://jiaowu.zhbit.com/default2.aspx
   User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36
```

恩呢，现在我们把Cookie清掉(或者等几分钟)，然后再访问下述链接：

jiaowu.zhbit.com/xs_main.aspx?xh=1102020

这时候，页面竟然自动跳回登陆页面了！当然一些其他的网站可能会弹出一个对话框说 "登陆超时"之类的东西！

小结下Http请求登陆的一个简单流程： 一般是登陆的时候：服务器通过Set-Cookie响应头，返回一个Cookie，浏览器默认保存这个Cookie， 后续访问相关页面的时候会带上这个Cookie，通过Cookie请求头来完成访问，如果没Cookie或者 Cookie过期，就提示用户没登陆，登陆超时，访问需要登陆之类的信息！

而我们使用HttpClient和HttpURLConnection其实也就是模拟这一个流程，登陆后拿到cookie 拿着它去发送请求： 关键代码如下： 获得**Cookie:conn.getHeaderField("Set-Cookie");** 请求时带上**Cookie:conn.setRequestProperty("Cookie",cookie);**

另外，除了这种设置请求头的方式外，还可以用另一种折衷的方法：**URL重写**： 就是在原先请求链接的基础上，加上一个...&sessionid=xxxxx这样的参数，然后由服务器来解析 判断！Get可以这么写，而Post写法如下：

```
HttpPut httpPut = new HttpPut(PUTACKCODE_URL);
httpPut.setHeader("Content-Type", "application/json; charset=UTF-8");
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("activation_code", actCode));
params.add(new BasicNameValuePair("session", new JSONObject().put("id", cookie).toString()));
entity.setContentType("application/json");
httpPut.setEntity(entity);
HttpResponse course_response = new DefaultHttpClient().execute(httpPut);
```

> 这里我们用的是JSON字符串的形式，接到请求时服务端取出session里的内容，然后做下查询即可~

5.使用HttpURLConnection发送PUT请求

> Put请求对于很多朋友来说可能有点陌生，毕竟我们平时接触的比较多的情况都是GET和POST， 一开始小猪也不知道，不过后来才发现和POST其实是差不多的，而且我们只需在POST的基础上改 点东西就可以使用了！而HttpClient也给我们提供了一个HttpPut的API， 下面贴下小猪自己项目中写的请求代码：

```
public  static  String  LoginByPut(Context mContext,  String mobile
```

本节小结：

> 好的，本节关于HttpUrlConnection的使用介绍就到这里，另外，HTTP这一小节大部分来自于小猪 以前写的一个小合集**Android**之**Http**通信，如果看过这个系列的可以跳过这节，大部分内容都是 一样的！嗯，就说这么多，谢谢~

7.1.4 Android HTTP请求方式:HttpClient

本节引言：

> 在上一节中我们对HttpURLConnection进行了学习，本节到第二种方式：HttpClient，尽管被Google 弃用了，但是我们我们平时也可以拿HttpClient来抓下包，配合Jsoup解析网页效果更佳！HttpClient 用于接收/发送Http请求/响应，但不缓存服务器响应，不执行HTML页面潜入的JS代码，不会对页面内容进行任何解析，处理！开始本节内容！

1.HttpClient使用流程

基本流程：

①创建HttpClient对象:HttpClient client = new DefaultHttpClient();
②发送Get请求的话,创建HttpGet对象;发送Post请求的话创建HttpPost对象
③设置请求参数:两者都可以用setParams(HttpParams);Post还可用setEnity(HttpEntity)方法
④调用httpClient对象的execute()方法发送请求,该方法会返回一个HttpResponse
⑤对返回的HttpResponse调用:getAllHeaders(),getHeaders(name)等方法可以获得服务器的响应头
调用getEntity()方法可以取得HttpEntity对象,改对象包装了服务器的响应内容

2.HttpClient使用示例

1）使用HttpClient发送GET请求

直接贴下简单的发送Get请求的代码：

```
public class MainActivity extends Activity implements OnClickLister

    private Button btnGet;
    private WebView wView;
    public static final int SHOW_DATA = 0X123;
    private String detail = "";

    private Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            if(msg.what == SHOW_DATA)
            {
                wView.loadDataWithBaseURL("",detail, "text/html","U
            }
        };
    };
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
        setView();
    }

    private void initView() {
        btnGet = (Button) findViewById(R.id.btnGet);
        wView = (WebView) findViewById(R.id.wView);
    }

    private void setView() {
        btnGet.setOnClickListener(this);
        wView.getSettings().setDomStorageEnabled(true);
    }
    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btnGet) {
            GetByHttpClient();
        }
    }
    private void GetByHttpClient() {
        new Thread()
        {
            public void run()
            {
                    try {
                        HttpClient httpClient = new DefaultHttpClie
                        HttpGet httpGet = new HttpGet("http://www.w
                        HttpResponse httpResponse = httpClient.exec
                        if (httpResponse.getStatusLine().getStatusC
                            HttpEntity entity = httpResponse.getEnt
                            detail = EntityUtils.toString(entity, "
                            handler.sendEmptyMessage(SHOW_DATA);
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            };
        }.start();
    }

}
```

运行截图

2:54

HttpClientDemo

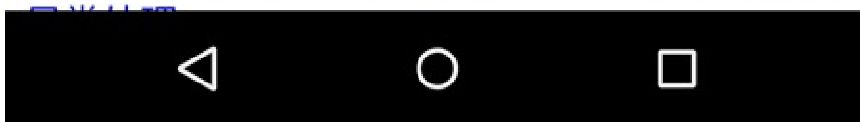
发送GET请求

Search w3cschool.cc: 搜索

首页 HTML CSS JavaScript jQuery jQueryUI Bootstrap jMobile EasyUI XML SQL MySQL SQLite PHP C# Python Java JSP Ruby 前端工具 Bug 反馈 更多......

Python 基础教程

Python 基础教程 Python 简介 Python 环境搭建 Python 中文编码 Python 基础语法 Python 变量类型 Python 运算符 Python 条件语句 Python 循环语句 Python While循环语句 Python for 循环语句 Python 循环嵌套 Python break 语句 Python continue 语句 Python pass 语句 Python 数字 Python 字符串 Python 列表(Lists) Python 元组 Python 字典(Dictionary) Python 日期和时间 Python 函数 Python 模块 Python 文件I/O Python



另外，如果是带有参数的GET请求的话，我们可以将参数放到一个List集合中，再对参数进行URL编码，最后和URL拼接下就好了：

```
List<BasicNameValuePair> params = new LinkedList<BasicNameValuePai
params.add(new BasicNameValuePair("user", "猪小弟"));
params.add(new BasicNameValuePair("pawd", "123"));
String param = URLEncodedUtils.format(params, "UTF-8");
HttpGet httpGet = new HttpGet("http://www.baidu.com"+"?"+param);
```

2）使用HttpClient发送POST请求

> POST请求比GET稍微复杂一点，创建完HttpPost对象后，通过NameValuePair集合来存储等待提交 的参数，并将参数传递到UrlEncodedFormEntity中，最后调用setEntity(entity)完成，HttpClient.execute(HttpPost)即可；这里就不写例子了，暂时没找到Post的网站，又不想 自己写个Servlet，So,直接贴核心代码吧~

核心代码:

```
private void PostByHttpClient(final String url)
{
    new Thread()
    {
        public void run()
        {
            try{
                HttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                List<NameValuePair> params = new ArrayList<NameValu
                params.add(new BasicNameValuePair("user", "猪大哥")
                params.add(new BasicNameValuePair("pawd", "123"));
                UrlEncodedFormEntity entity = new UrlEncodedFormEnt
                httpPost.setEntity(entity);
                HttpResponse httpResponse =  httpClient.execute(htt
                if (httpResponse.getStatusLine().getStatusCode() ==
                    HttpEntity entity2 = httpResponse.getEntity();
                    detail = EntityUtils.toString(entity2, "utf-8"
                    handler.sendEmptyMessage(SHOW_DATA);
                }
            }catch(Exception e){e.printStackTrace();}
        };
    }.start();
}
```

3.HttpClient抓数据示例(教务系统数据抓取)

> 其实关于HttpClient的例子有很多，比如笔者曾经用它来抓学校教务系统上学生的课程表: 这就涉及到Cookie，模拟登陆的东西，说到抓数据(爬虫)，一般我们是搭配着JSoup来解析 抓到数据的，有兴趣可以自己查阅相关资料，这里贴下笔者毕设app里获取网页部分的关键 代码！大家可以体会下：

HttpClient可以通过下述代码获取与设置Cookie： **HttpResponse loginResponse = new DefaultHttpClient().execute(getLogin);** 获得Cookie:**cookie = loginResponse.getFirstHeader("Set-Cookie").getValue();** 请求时带上Cookie:**httpPost.setHeader("Cookie", cookie);**

```
//获得链接,模拟登录的实现:
public int getConnect(String user, String key) throws Exception {
    // 先发送get请求 获取cookie值和__ViewState值
    HttpGet getLogin = new HttpGet(true_url);
    // 第一步:主要的HTML:
    String loginhtml = "";
    HttpResponse loginResponse = new DefaultHttpClient().execute(ge
    if (loginResponse.getStatusLine().getStatusCode() == 200) {
        HttpEntity entity = loginResponse.getEntity();
        loginhtml = EntityUtils.toString(entity);
        // 获取响应的cookie值
        cookie = loginResponse.getFirstHeader("Set-Cookie").getValu
        System.out.println("cookie= " + cookie);
    }

    // 第二步:模拟登录
    // 发送Post请求,禁止重定向
    HttpPost httpPost = new HttpPost(true_url);
    httpPost.getParams().setParameter(ClientPNames.HANDLE_REDIRECTS

    // 设置Post提交的头信息的参数
    httpPost.setHeader("User-Agent",
            "Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11
    httpPost.setHeader("Referer", true_url);
    httpPost.setHeader("Cookie", cookie);

    // 设置请求数据
    List<NameValuePair> params = new ArrayList<NameValuePair>();

    params.add(new BasicNameValuePair("__VIEWSTATE",
            getViewState(loginhtml)));// __VIEWSTATE参数, 如果变化可以
    params.add(new BasicNameValuePair("Button1", ""));
    params.add(new BasicNameValuePair("hidPdrs", ""));
    params.add(new BasicNameValuePair("hidsc", ""));
    params.add(new BasicNameValuePair("lbLanguage", ""));
    params.add(new BasicNameValuePair("RadioButtonList1", "%D1%A7%C
    params.add(new BasicNameValuePair("txtUserName", user));
    params.add(new BasicNameValuePair("TextBox2", key));
    params.add(new BasicNameValuePair("txtSecretCode", "")); // (

    // 设置编码方式,响应请求,获取响应状态码:
    httpPost.setEntity(new UrlEncodedFormEntity(params, "gb2312"));
    HttpResponse response = new DefaultHttpClient().execute(httpPos
    int Status = response.getStatusLine().getStatusCode();
    if(Status == 200)return Status;
    System.out.println("Status= " + Status);

    // 重定向状态码为302
    if (Status == 302 || Status == 301) {
        // 获取头部信息中Location的值
        location = response.getFirstHeader("Location").getValue();
        System.out.println(location);
```

```
            // 第三步:获取管理信息的主页面
            // Get请求
            HttpGet httpGet = new HttpGet(ip_url + location);// 带上loc
            httpGet.setHeader("Referer", true_url);
            httpGet.setHeader("Cookie", cookie);

            // 主页的html
            mainhtml = "";
            HttpResponse httpResponseget = new DefaultHttpClient()
                    .execute(httpGet);
            if (httpResponseget.getStatusLine().getStatusCode() == 200)
                HttpEntity entity = httpResponseget.getEntity();
                mainhtml = EntityUtils.toString(entity);
            }

        }
        return Status;
    }
```

4.使用HttpPut发送Put请求

示例代码如下：

```
public static int PutActCode(String actCode, String licPlate, Conte
    int resp = 0;
    String cookie = (String) SPUtils.get(mContext, "session", "");
    HttpPut httpPut = new HttpPut(PUTACKCODE_URL);
    httpPut.setHeader("Cookie", cookie);
    try {

        List<NameValuePair> params = new ArrayList<NameValuePair>(
        params.add(new BasicNameValuePair("activation_code", actCod
        params.add(new BasicNameValuePair("license_plate", licPlate
        httpPut.setEntity(new UrlEncodedFormEntity(params, "UTF-8")
        HttpResponse course_response = new DefaultHttpClient().exec
        if (course_response.getStatusLine().getStatusCode() == 200)
            HttpEntity entity2 = course_response.getEntity();
            JSONObject jObject = new JSONObject(EntityUtils.toStrin
            resp = Integer.parseInt(jObject.getString("status_code"
            return resp;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return resp;
}
```

本节小结：

> 好的，本节关于Android HTTP的第二种请求方式:HttpClient就到这里，下节开始我们来学习XML以及Json的解析，本节就到这里，谢谢~

7.2.1 Android XML数据解析

本节引言：

> 前面两节我们对Android内置的Http请求方式：HttpURLConnection和HttpClient，本来以为OkHttp 已经集成进来了，然后想讲解下Okhttp的基本用法，后来发现还是要导第三方，算了，放到进阶部分 吧，而本节我们来学习下Android为我们提供的三种解析XML数据的方案！他们分别是：SAX,DOM,PULL三种解析方式，下面我们就来对他们进行学习！

1.XML数据要点介绍

首先我们来看看XML数据的一些要求以及概念：

XML格式数据的简单理解

全名叫做可扩展标记语言,类似于HTML(超文本标记),这里不深究他的定义,只介绍他是用来做什么的!其实我们更多的时候是把xml用来存储数据,可以看作一个微型的数据库,比如我们前面学的SharedPreference就是使用xml文件来保存配置信息的,而我们的SQLite其实底层也是一个xml文件;而在网络应用方面通常作为信息的载体,我们通常把数据包装成xml来传递

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
    <person id = "11">
        <name>Coder-pig</name>
        <age>20</age>
    </person>
    <person id = "13">
        <name>Jay</name>
        <age>20</age>
    </person>
</persons>
```

对应结构

文档开始
开始元素(persons)
文本结点(空白文本) 开始元素(person) 属性
文本结点(空白文本) 开始元素(name) 文本结点 结束元素
文本结点(空白文本) 开始元素(age) 文本结点 结束元素
文本结点(空白文本) 结束元素
...
结束元素(persons)
文档结束

ps:上面就是简单的定义一个存储person对象的xml文件的编码;要注意一点哦,外面的空白区域也是文本结点哦！

2.三种解析XML方法的比较

SAX,DOM,PULL解析xml的比较

SAX解析XML:

对文档进行顺序扫描,当扫描到文档(document)开始与结束、元素(element)开始与结束、文档(document)结束等地方时通知事件处理函数，由事件处理函数做相应动作，然后继续同样的扫描，直至文档结束。解析速度快,占用的内存也少
每需要解析一类xml,就需要编写新的适合该类XML的处理类,用起来还是有点麻烦的!采用的是流式解析,解析是同步的;读到哪就处理哪！

Dom解析XML:

先把xml文档都读到内存中，然后再用DOM API来访问树形结构，并获取数据。这个写起来很简单，但是很消耗内存.加入读取的数据量大,手机内存不够的话,可能导致手机死机!不建议在Android设备中使用,解析简单的xml文件倒可以！
常用的五个接口与类:Document,Element,Node,NodeList,DOMParser
Dom是整个文件解析到内存中,供用户需要的结点信息,支持随机访问

pull解析XML:

XML pull提供了开始元素和结束元素。当某个元素开始时，我们可以调用parser．nextText0从XML文档中提取所有字符数据。当解释到一个文档结束时，自动生成EndDocument
常用接口和类:XmlPullParser,XmlSrealizer,XmlPullParserFactory
和SAX差不多,代码没那么实现较为简单,非常适合移动设备,Android系统内置pull解析器,而且Android系统内部默认使用pull来解析xml文件,如果需要在J2EE或者其他使用pull需要导入两个包,后面给出下载

3.SAX解析XML数据

SAX解析xml文件

SAX是一个解析速度快且占用内存少的xml解析器,非常适合用于Android等移动设备;SAX解析xml文件采用的是事件驱动;也就是不需要解析整个文档,而是在解析文档的过程中,判断读到的字符是否符合xml语法的某部分(文件开头,文档结束,或者标签开头与标签结束时)符合的话就会触发事件(回调方法)而这些方法都定义在ContentHandler接口中,而ContentHanlder是一个接口,使用起来有些不方便,而Android很贴心地为我们提供了一个帮助类:DefaultHandler,只需要继承这个类,重写对应的方法即可

可供重写的方法如下：

startDocument():	当读取到文档开始标志时触发,通常在这里完成一些初始化操作
endDocument():	文档结束部分,在这里完成一些善后工作
startElement(namespaceURI,localName,qName,atts):	参数依次为命名空间;不带命名空间的前缀标签名;带命名控件前缀名的标签,通过atts可以得到所有的属性名与相应的值；SAX中一个重要的特点就是它的流式处理，当遇到一个标签的时候，它并不会纪录下以前所碰到的标签，也就是说，在startElement()方法中，所有你所知道的信息，就是标签的名字和属性，至于标签的嵌套结构，上层标签的名字，是否有子元属等等其它与结构相关的信息，都是不得而知的，都需要你的程序来完成。这使得SAX在编程处理上没有DOM来得那么方便。
endElement(uri,localName,name)	在遇到结束标签的时候，调用这个方法
characters(ch,start,length)	这个方法用来处理在XML文件中读到的内容，第一个参数用于存放文件的内容，后面两个参数是读到的字符串在这个数组中的起始位置和长度，使用new String(ch,start,length)就可以获取内容。

核心代码：

SAX解析类：**SaxHelper.java**：

```
/**
 * Created by Jay on 2015/9/8 0008.
 */
public class SaxHelper extends DefaultHandler {
    private Person person;
    private ArrayList<Person> persons;
    //当前解析的元素标签
    private String tagName = null;

    /**
     * 当读取到文档开始标志是触发，通常在这里完成一些初始化操作
     */
    @Override
    public void startDocument() throws SAXException {
        this.persons = new ArrayList<Person>();
        Log.i("SAX", "读取到文档头,开始解析xml");
    }

    /**
     * 读到一个开始标签时调用,第二个参数为标签名,最后一个参数为属性数组
     */
    @Override
    public void startElement(String uri, String localName, String q
                             Attributes attributes) throws SAXExcep
        if (localName.equals("person")) {
            person = new Person();
            person.setId(Integer.parseInt(attributes.getValue("id")
            Log.i("SAX", "开始处理person元素~");
        }
        this.tagName = localName;
    }

    /**
     * 读到到内容,第一个参数为字符串内容,后面依次为起始位置与长度
     */

    @Override
    public void characters(char[] ch, int start, int length)
            throws SAXException {
        //判断当前标签是否有效
        if (this.tagName != null) {
            String data = new String(ch, start, length);
            //读取标签中的内容
            if (this.tagName.equals("name")) {
                this.person.setName(data);
                Log.i("SAX", "处理name元素内容");
            } else if (this.tagName.equals("age")) {
                this.person.setAge(Integer.parseInt(data));
```

```
                Log.i("SAX", "处理age元素内容");
            }

        }

    }

    /**
     * 处理元素结束时触发,这里将对象添加到结合中
     */
    @Override
    public void endElement(String uri, String localName, String qNa
            throws SAXException {
        if (localName.equals("person")) {
            this.persons.add(person);
            person = null;
            Log.i("SAX", "处理person元素结束~");
        }
        this.tagName = null;
    }

    /**
     * 读取到文档结尾时触发,
     */
    @Override
    public void endDocument() throws SAXException {
        super.endDocument();
        Log.i("SAX", "读取到文档尾,xml解析结束");
    }

    //获取persons集合
    public ArrayList<Person> getPersons() {
        return persons;
    }

}
```

然后我们在MainActivity.java中写上写上这样一个方法，然后要解析XML的时候调用下 就好了~

```
private ArrayList<Person> readxmlForSAX() throws Exception {
    //获取文件资源建立输入流对象
    InputStream is = getAssets().open("person1.xml");
    //①创建XML解析处理器
    SaxHelper ss = new SaxHelper();
    //②得到SAX解析工厂
    SAXParserFactory factory = SAXParserFactory.newInstance();
    //③创建SAX解析器
    SAXParser parser = factory.newSAXParser();
    //④将xml解析处理器分配给解析器,对文档进行解析,将事件发送给处理器
    parser.parse(is, ss);
    is.close();
    return ss.getPersons();
}
```

一些其他的话：

嗯，对了，忘记给大家说下我们是定义下面这样一个person1.xml文件，然后放到assets目录下的！ 文件内容如下：**person1.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
    <person id = "11">
        <name>SAX解析</name>
        <age>18</age>
    </person>
    <person id = "13">
        <name>XML1</name>
        <age>43</age>
    </person>
</persons>
```

我们是把三种解析方式都糅合到一个demo中，所以最后才贴全部的效果图，这里的话，贴下打印的Log， 相信大家会对SAX解析XML流程更加明了：

```
I/SAX： 读取到文档头,开始解析xml
I/SAX： 开始处理person元素~
I/SAX： 处理name元素内容
I/SAX： 处理age元素内容
I/SAX： 处理person元素结束~
I/SAX： 开始处理person元素~
I/SAX： 处理name元素内容
I/SAX： 处理age元素内容
I/SAX： 处理person元素结束~
I/SAX： 读取到文档尾,xml解析结束
```

另外，外面的空白文本也是文本节点哦！解析的时候也会走这些节点！

4.DOM解析XML数据

Dom解析XML文件

Dom解析XML文件时会将XML文件的所有内容以文档树的形式存放到内存中,然后允许您使用DOM API遍历XML树、检索所需的数据。使用DOM操作XML的代码看起来是比较直观的，并且在编码方面比基于SAX的实现更加简单。但是，因为DOM需要将XML文件的所有内容以文档树方式存放在内存中，所以内存的消耗比较大，特别对于运行Android的移动设备来说，因为设备的资源比较宝贵，所以建议还是采用SAX来解析XML文件，当然，如果XML文件的内容比较小采用DOM也是可行的。

先了解一下Dom的一些api

类	说明
DocumentBuilderFactory (解析器工厂类)	创建方法: DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder (解析器类)	创建方法: 通过解析器工厂类来获得 DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
Document (文档树模型)	将要解析的xml文件读入Dom解析器 Document doc = dbBuilder.parse(context.getAssets().open("persons2.xml"));

ps:Document对象代表了一个xml文档的模型树,所有的其他Node都以一定的顺序包含在Document对象之内，排列成一个树状结构，以后对XML文档的所有操作都与解析器无关，

类	说明
NodeList (结点列表类)	代表一个包含一个或者多个Node的列表,可看作数组,有以下两个方法: item(index):返回集合的第index个Node项　getLength():列表的节点数
Node (结点类)	Dom中最基本的对象,代表文档树中的抽象结点,很少会直接使用的;通常是调用他的子对象Element,Attr,Text等
Element (元素类)	Node最主要的自对象,再元素中可以包含属性,因此有取属性的方法: getAttribute()获得属性值;getTagName():元素的名称;
Attr (属性类)	代表某个元素的属性，虽然Attr继承自Node接口，但因为Attr是包含在Element中的，但并不能将其看做是Element的子对象，因为Attr并不是DOM树的一部分

核心代码：

DomHelper.java

```
/**
 * Created by Jay on 2015/9/8 0008.
 */
public class DomHelper {
    public static ArrayList<Person> queryXML(Context context)
    {
        ArrayList<Person> Persons = new ArrayList<Person>();
        try {
            //①获得DOM解析器的工厂示例:
            DocumentBuilderFactory dbFactory = DocumentBuilderFacto
            //②从Dom工厂中获得dom解析器
            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilde
            //③把要解析的xml文件读入Dom解析器
            Document doc = dbBuilder.parse(context.getAssets().oper
            System.out.println("处理该文档的DomImplemention对象=" + d
            //④得到文档中名称为person的元素的结点列表
            NodeList nList = doc.getElementsByTagName("person");
            //⑤遍历该集合,显示集合中的元素以及子元素的名字
            for(int i = 0;i < nList.getLength();i++)
            {
                //先从Person元素开始解析
                Element personElement = (Element) nList.item(i);
                Person p = new Person();
                p.setId(Integer.valueOf(personElement.getAttribute(

                //获取person下的name和age的Note集合
                NodeList childNoList = personElement.getChildNodes(
                for(int j = 0;j < childNoList.getLength();j++)
                {
                    Node childNode = childNoList.item(j);
                    //判断子note类型是否为元素Note
                    if(childNode.getNodeType() == Node.ELEMENT_NODE
                    {
                        Element childElement = (Element) childNode
                        if("name".equals(childElement.getNodeName()
                            p.setName(childElement.getFirstChild()
                        else if("age".equals(childElement.getNodeNa
                            p.setAge(Integer.valueOf(childElement.g
                    }
                }
                Persons.add(p);
            }
        } catch (Exception e) {e.printStackTrace();}
        return Persons;
    }
}
```

代码分析：

> 从代码我们就可以看出DOM解析XML的流程，先整个文件读入Dom解析器，然后形成一棵树， 然后我们可以遍历节点列表获取我们需要的数据!

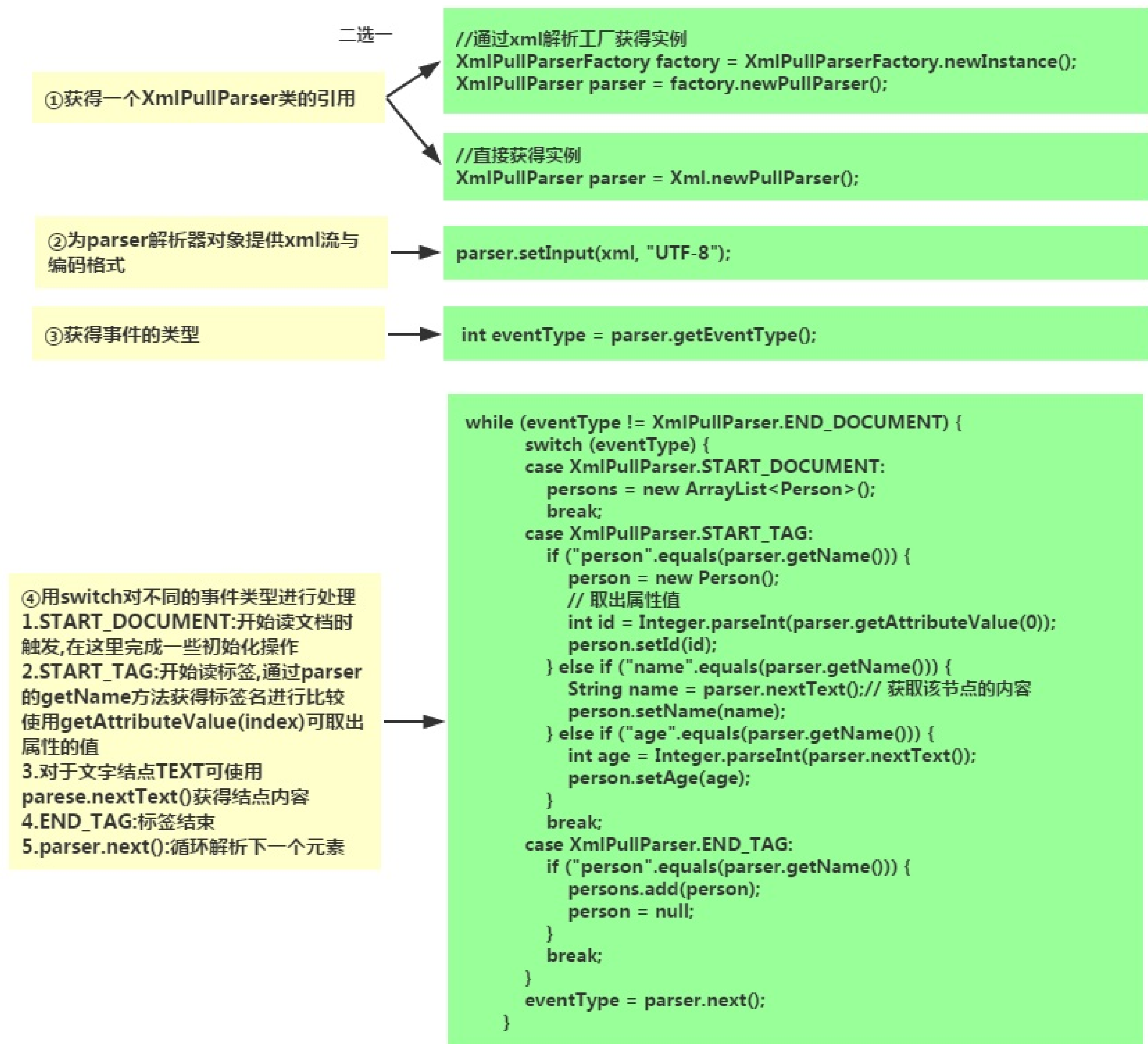
5.PULL解析XML数据

使用Pull解析xml

简单介绍

除了前面的SAX和DOM解析XML文件外,Android系统其实内置了Pull解析器用来解析xml文件,比如我们前面学到的SharedPreference就是使用的内置pull解析配置文件的！他的使用和SAX相似,都是采用事件驱动来完成xml的解析的;
而pull的编码较为简单,只需处理开始与结束事件,通常是使用switch语句,根据事件的不同类型,匹配不同的处理方式,
有五种事件:START_DOCUMENT;START_TAG;TEXT;END_TAG;END_DOCUMENT
XML pull提供了开始元素和结束元素。当某个元素开始时，可以调用parser.nextText从XML文档中提取所有字符数据。当解析到一个文档结束时，自动生成EndDocument事件。在PULL解析过程中返回的是数字，且我们需要自己获取产生的事件然后做相应的操作，而不像SAX那样由处理器触发一种事件的方法，执行我们的代码：
读取到xml的声明返回 START_DOCUMENT； 结束返回 END_DOCUMENT ； 开始标签返回 START_TAG；
结束标签返回 END_TAG； 文本返回 TEXT。

使用**PULL**解析**XML**数据的流程：


使用Pull解析xml文件流程:
二选一
①获得一个XmlPullParser类的引用
//通过xml解析工厂获得实例
XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
XmlPullParser parser = factory.newPullParser();
//直接获得实例
XmlPullParser parser = Xml.newPullParser();
②为parser解析器对象提供xml流与编码格式
parser.setInput(xml, "UTF-8");
③获得事件的类型
int eventType = parser.getEventType();
④用switch对不同的事件类型进行处理
1.START_DOCUMENT:开始读文档时触发,在这里完成一些初始化操作
2.START_TAG:开始读标签,通过parser的getName方法获得标签名进行比较使用getAttributeValue(index)可取出属性的值
3.对于文字结点TEXT可使用parese.nextText()获得结点内容
4.END_TAG:标签结束
5.parser.next():循环解析下一个元素
while (eventType != XmlPullParser.END_DOCUMENT) {
switch (eventType) {
case XmlPullParser.START_DOCUMENT:
persons = new ArrayList<Person>();
break;
case XmlPullParser.START_TAG:
if ("person".equals(parser.getName())) {
person = new Person();
// 取出属性值
int id = Integer.parseInt(parser.getAttributeValue(0));
person.setId(id);
} else if ("name".equals(parser.getName())) {
String name = parser.nextText();// 获取该节点的内容
person.setName(name);
} else if ("age".equals(parser.getName())) {
int age = Integer.parseInt(parser.nextText());
person.setAge(age);
}
break;
case XmlPullParser.END_TAG:
if ("person".equals(parser.getName())) {
persons.add(person);
person = null;
}
break;
}
eventType = parser.next();
}

核心代码：

```
public static ArrayList<Person> getPersons(InputStream xml)throws E
{
    //XmlPullParserFactory pullPaser = XmlPullParserFactory.newInst
    ArrayList<Person> persons = null;
    Person person = null;
    // 创建一个xml解析的工厂
    XmlPullParserFactory factory = XmlPullParserFactory.newInstance
    // 获得xml解析类的引用
    XmlPullParser parser = factory.newPullParser();
    parser.setInput(xml, "UTF-8");
    // 获得事件的类型
    int eventType = parser.getEventType();
    while (eventType != XmlPullParser.END_DOCUMENT) {
        switch (eventType) {
        case XmlPullParser.START_DOCUMENT:
            persons = new ArrayList<Person>();
            break;
        case XmlPullParser.START_TAG:
            if ("person".equals(parser.getName())) {
                person = new Person();
                // 取出属性值
                int id = Integer.parseInt(parser.getAttributeValue(
                person.setId(id);
            } else if ("name".equals(parser.getName())) {
                String name = parser.nextText();// 获取该节点的内容
                person.setName(name);
            } else if ("age".equals(parser.getName())) {
                int age = Integer.parseInt(parser.nextText());
                person.setAge(age);
            }
            break;
        case XmlPullParser.END_TAG:
            if ("person".equals(parser.getName())) {
                persons.add(person);
                person = null;
            }
            break;
        }
        eventType = parser.next();
    }
    return persons;
}
```

使用**Pull**生成**xml**数据的流程:


使用Pull生成xml文件流程:
①创建XmlSerializer(xml序列化类)的实例
XmlSerializer serializer = Xml.newSerializer();
②为XmlSerializer设置输出流与编码格式
serializer.setOutput(out,"UTF-8");
③为XmlSerializer设置xml的编码格式
serializer.startDocument("UTF-8",true);
④设置根元素
serializer.startTag(null, "persons");
④使用增强for循环遍历persons集合中所有的元素,同时依次写入标签与属性
for(Person p:persons)
{
serializer.startTag(null, "person");
serializer.attribute(null, "id",p.getId()+"");
serializer.startTag(null, "name");
serializer.text(p.getName());
serializer.endTag(null, "name");
serializer.startTag(null, "age");
serializer.text(p.getAge()+"");
serializer.endTag(null, "age");
serializer.endTag(null, "person");
}
⑤设置跟完结元素
serializer.endTag(null, "persons");
⑥结束文档编写
serializer.endDocument();
⑦调用flush()将内存中的数据写入文件中关闭输出流
out.flush();
out.close();

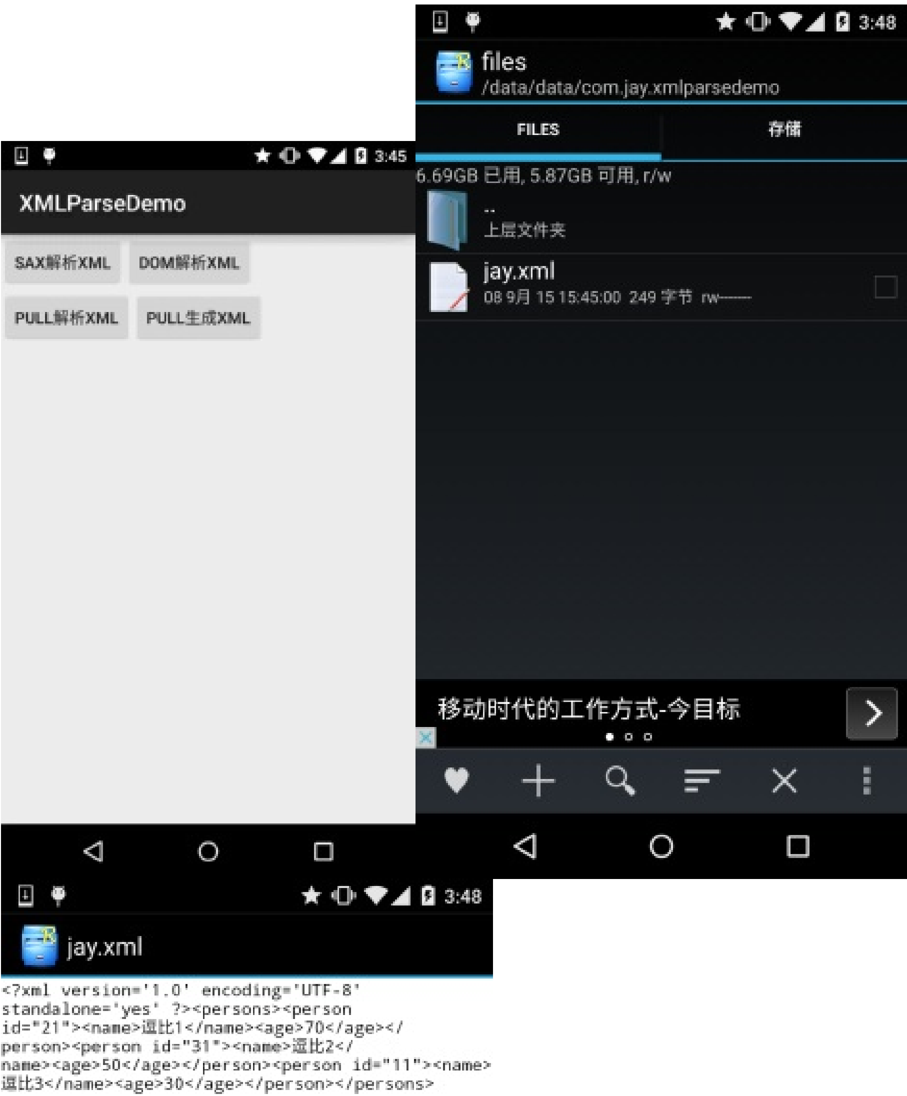
核心代码：

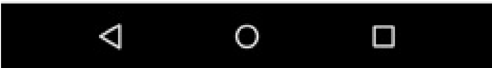
```
public static void save(List<Person> persons, OutputStream out) thr
    XmlSerializer serializer = Xml.newSerializer();
    serializer.setOutput(out, "UTF-8");
    serializer.startDocument("UTF-8", true);
    serializer.startTag(null, "persons");
    for (Person p : persons) {
        serializer.startTag(null, "person");
        serializer.attribute(null, "id", p.getId() + "");
        serializer.startTag(null, "name");
        serializer.text(p.getName());
        serializer.endTag(null, "name");
        serializer.startTag(null, "age");
        serializer.text(p.getAge() + "");
        serializer.endTag(null, "age");
        serializer.endTag(null, "person");
    }

    serializer.endTag(null, "persons");
    serializer.endDocument();
    out.flush();
    out.close();
}
```

6.代码示例下载：

运行效果图：





代码下载： **XMLParseDemo.zip**：下载 XMLParseDemo.zip

本节小结：

> 本节介绍了Android中三种常用的XML解析方式，DOM，SAX和PULL，移动端我们建议用后面这 两种，而PULL用起来更加简单，这里就不多说了，代码是最好的老师~本节就到这里，下节我们 来学习Android为我们提供的扣脚Json解析方式！谢谢~

7.2.2 Android JSON数据解析

本节引言：

> 相信大家肯定对JSON不陌生吧，我们和服务器交互一般用得较多的数据传递方式都是Json字符串的形式， 保存对象，我们也可以写成一个Json字符串然后存储！解析Json不知道你用的是Gson，Fastjson，jackson 等，不过本节我们并不会去用这些第三方的解析库，而是使用Android自带的扣脚Json解析器来解析Json！ 好的，开始本节内容！

1.Json概念以及与XML的比较

1）Json是什么？

> 答：JavaScript Object Natation, 一种轻量级的数据交换格式, 与XML一样, 广泛被采用的客户端和服务端交互的解决方案！具有良好的可读和便于快速编写的特性。

2）Json与XML的比较：

> - JSON和XML的数据可读性基本相同;
> - JSON和XML同样拥有丰富的解析手段
> - JSON相对于XML来讲，数据的体积小
> - JSON与JavaScript的交互更加方便
> - JSON对数据的描述性比XML较差
> - JSON的速度要远远快于XML
>
> PS:上述来自于百度~简单点说Json的优点：体积小， 节省流量， 只是不如XML直观， 可读性 稍微差一点而已！

3）Json的格式规范：

> 就像协议一样，肯定是有一套规范的， 毕竟双方都是通过Json字符串来传递数据， 语法规则如下： 数据在名称/值对中；数据由逗号分隔；花括号保存对象；方括号保存数组； 而Json数据的书写格式：名称/值对 比如：**"person"**：**"coder-pig"** 比如一个简单的Json字符串：

```
[
    { "id":"1","name":"基神","age":"18" },
    { "id":"2","name":"B神","age":"18"  },
    { "id":"3","name":"曹神","age":"18" }
]
```

> 我们除了解析Json还可以自己拼接Json，当然如果你自己拼了一个Json字符串又不知道对不对， 可以百度随便找一个校验工具，比如：http://www.runoob.com/jsontool把Json字符串贴上去，校验下就好！

2.Android给我们提供的扣脚Json解析类

> 这些API都存在于org.json包下，而我们用到的类有下面这些：
>
> - **JSONObject**： Json对象，可以完成Json字符串与Java对象的相互转换
> - **JSONArray**： Json数组，可以完成Json字符串与Java集合或对象的相互转换
> - **JSONStringer**： Json文本构建类，这个类可以帮助快速和便捷的创建JSON text， 每个JSONStringer实体只能对应创建一个JSON text
> - **JSONTokener**：Json解析类
> - **JSONException**：Json异常

3.代码示例：解析Json字符串：

PS:这里我们就不另外写servlet或者请求网站，直接把Json写到字符串中来解析，模拟下就算了！

1)简单的Json字符串解析示例：

我们解析的是上面这个简单的Json，首先我们来写一个POJO类：

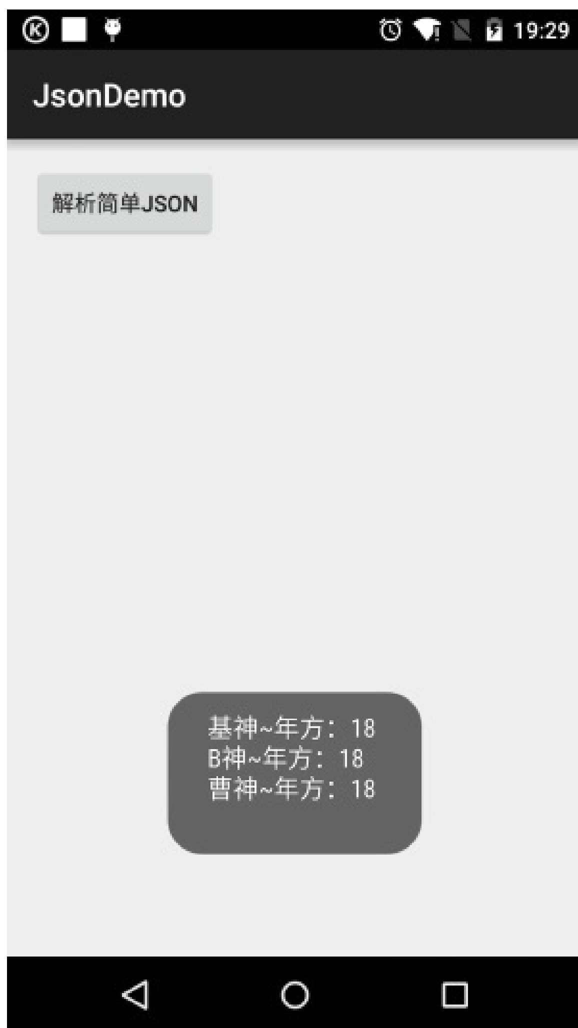
Person.java：

```
/**
 * Created by Jay on 2015/9/8 0008.
 */
public class Person {
    private String id;
    private String name;
    private String age;
    public void setId(String id){
        this.id = id;
    }
    public String getId(){
        return this.id;
    }
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }
    public void setAge(String age){
        this.age = age;
    }
    public String getAge(){
        return this.age;
    }
    @Override
    public String toString() {
        return this.name + "~年方：" + this.age;
    }
}
```

写一个解析上述Json字符串的方法：

```
private void parseEasyJson(String json){
    persons = new ArrayList<Person>();
    try{
        JSONArray jsonArray = new JSONArray(json);
        for(int i = 0;i < jsonArray.length();i++){
            JSONObject jsonObject = (JSONObject) jsonArray.get(i);
            Person person = new Person();
            person.setId(i+"");
            person.setName(jsonObject.getString("name"));
            person.setAge(jsonObject.getString("age"));
            persons.add(person);
        }
    }catch (Exception e){e.printStackTrace();}
}
```

运行效果图：



嘿嘿，很简单是吧，接下来我们找一个复杂点的！

2)复杂的Json字符串解析示例：

如果是这样的Json字符串呢？

{"ch":[{"names":"北理工","data":[2,2,1,1,1,1],"times":[10,11,13,13,21,23]},{"names":"北师大","data":[2,2,1,1,1,1],"times":[10,11,13,13,21,23]}]}

呵呵，那就需要我们一步步来扣数据了：

解析代码如下：

```
private void parseDiffJson(String json) {
    try {
        JSONObject jsonObject1 = new JSONObject(json);
        Log.e("Json", json);
        JSONArray jsonArray = jsonObject1.getJSONArray("ch");
        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = (JSONObject) jsonArray.get(i);
            //取出name
            String sname = jsonObject.getString("names");
            JSONArray jarray1 = jsonObject.getJSONArray("data");
            JSONArray jarray2 = jsonObject.getJSONArray("times");
            Log.e("Json", sname);
            Log.e("Json", jarray1.toString());
            Log.e("Json", jarray2.toString());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

看下打印的**Log**:

```
E/Json： {"ch":[{"names":"北理工","data":[2,2,1,1,1,1],"times":[10,11,13,13,21,23]},{"names":"北
E/Json： 北理工
E/Json： [2,2,1,1,1,1]
E/Json： [10,11,13,13,21,23]
E/Json： 北师大
E/Json： [2,2,1,1,1,1]
E/Json： [10,11,13,13,21,23]
```

当然还有一层，有兴趣你就自己扣...

本节小结：

> 好的，使用Android给我们提供的扣脚Json解析类果然要慢慢扣，当然你也可以将解析的过程 反过来，自己拼接JSON，时间关系，这里就慢慢拼接了，哈哈，当然进阶部分我们 学习了第三方的一些Json解析库就轻松多了，~好的，本节就到这里，谢谢~

7.3.1 Android 文件上传

本节引言

> 本节和下一节文件下载一样，慎入...现在实际开发涉及文件上传不会自己写上传代码，一般 会集成第三网络库来做图片上传，比如android-async-http，okhttp等，另外还有七牛也提供 了下载和上传的API，喜欢的可以去官网查看相关的API文档！本节的话有兴趣看看就好！

1.项目用到的图片上传的关键方法：

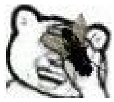
思前想后，还是决定先贴下公司项目中用到的图片上传的核心方法，这里用到一个第三方的库：android-async-http.jar，自己到github下下这个库~然后调用一下下面的方法即可，自己改下url!

上传图片的核心方法如下：

```
private void sendImage(Bitmap bm)
{
    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    bm.compress(Bitmap.CompressFormat.PNG, 60, stream);
    byte[] bytes = stream.toByteArray();
    String img = new String(Base64.encodeToString(bytes, Base64.DEF
    AsyncHttpClient client = new AsyncHttpClient();
    RequestParams params = new RequestParams();
    params.add("img", img);
    client.post("http:xxx/postIcon", params, new AsyncHttpResponseH
        @Override
        public void onSuccess(int i, Header[] headers, byte[] bytes
            Toast.makeText(MainActivity.this, "Upload Success!", To

        }
        @Override
        public void onFailure(int i, Header[] headers, byte[] bytes
            Toast.makeText(MainActivity.this, "Upload Fail!", Toast
        }
    });
}
```

2.使用HttpConnection上传文件：

简直卧槽...各种设置，各种麻烦...还是建议用1的方法吧，当然，实在太闲可以看看，有轮子可用还是先别自己造轮子了...

```
public class SocketHttpRequester
{
    /**
     * 发送xml数据
     * @param path 请求地址
     * @param xml xml数据
     * @param encoding 编码
     * @return
     * @throws Exception
     */
    public static byte[] postXml(String path, String xml, String en
        byte[] data = xml.getBytes(encoding);
        URL url = new URL(path);
        HttpURLConnection conn = (HttpURLConnection)url.openConnect
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);
        conn.setRequestProperty("Content-Type", "text/xml; charset=
        conn.setRequestProperty("Content-Length", String.valueOf(da
        conn.setConnectTimeout(5 * 1000);
        OutputStream outStream = conn.getOutputStream();
        outStream.write(data);
        outStream.flush();
        outStream.close();
        if(conn.getResponseCode()==200){
            return readStream(conn.getInputStream());
        }
        return null;
    }

    /**
     * 直接通过HTTP协议提交数据到服务器,实现如下面表单提交功能:
     *   <FORM METHOD=POST ACTION="http://192.168.0.200:8080/ssi/fi
            <INPUT TYPE="text" NAME="name">
            <INPUT TYPE="text" NAME="id">
            <input type="file" name="imagefile"/>
            <input type="file" name="zip"/>
         </FORM>
     * @param path 上传路径(注：避免使用localhost或127.0.0.1这样的路径测
     *                  因为它会指向手机模拟器，你可以使用http://www.bai
     * @param params 请求参数 key为参数名,value为参数值
     * @param file 上传文件
     */
    public static boolean post(String path, Map<String, String> par
    {
        //数据分隔线
        final String BOUNDARY = "---------------------------7da2137
        //数据结束标志"---------------------------7da2137580612--"
        final String endline = "--" + BOUNDARY + "--/r/n";
```

```
        //下面两个for循环都是为了得到数据长度参数，依据表单的类型而定
        //首先得到文件类型数据的总长度(包括文件分割线)
        int fileDataLength = 0;
        for(FormFile uploadFile : files)
        {
            StringBuilder fileExplain = new StringBuilder();
            fileExplain.append("--");
            fileExplain.append(BOUNDARY);
            fileExplain.append("/r/n");
            fileExplain.append("Content-Disposition: form-data;name
            fileExplain.append("Content-Type: "+ uploadFile.getCont
            fileExplain.append("/r/n");
            fileDataLength += fileExplain.length();
            if(uploadFile.getInStream()!=null){
                fileDataLength += uploadFile.getFile().length();
            }else{
                fileDataLength += uploadFile.getData().length;
            }
        }
        //再构造文本类型参数的实体数据
        StringBuilder textEntity = new StringBuilder();
        for (Map.Entry<String, String> entry : params.entrySet())
        {
            textEntity.append("--");
            textEntity.append(BOUNDARY);
            textEntity.append("/r/n");
            textEntity.append("Content-Disposition: form-data; name
            textEntity.append(entry.getValue());
            textEntity.append("/r/n");
        }

        //计算传输给服务器的实体数据总长度(文本总长度+数据总长度+分隔符)
        int dataLength = textEntity.toString().getBytes().length +

        URL url = new URL(path);
        //默认端口号其实可以不写
        int port = url.getPort()==-1 ? 80 : url.getPort();
        //建立一个Socket链接
        Socket socket = new Socket(InetAddress.getByName(url.getHos
        //获得一个输出流（从Android流到web）
        OutputStream outStream = socket.getOutputStream();
        //下面完成HTTP请求头的发送
        String requestmethod = "POST "+ url.getPath()+" HTTP/1.1/r/
        outStream.write(requestmethod.getBytes());
        //构建accept
        String accept = "Accept: image/gif, image/jpeg, image/pjpeg
        outStream.write(accept.getBytes());
        //构建language
        String language = "Accept-Language: zh-CN/r/n";
        outStream.write(language.getBytes());
        //构建contenttype
        String contenttype = "Content-Type: multipart/form-data; bo
```

```
        outStream.write(contenttype.getBytes());
        //构建contentlength
        String contentlength = "Content-Length: "+ dataLength + "/r
        outStream.write(contentlength.getBytes());
        //构建alive
        String alive = "Connection: Keep-Alive/r/n";
        outStream.write(alive.getBytes());
        //构建host
        String host = "Host: "+ url.getHost() +":"+ port +"/r/n";
        outStream.write(host.getBytes());
        //写完HTTP请求头后根据HTTP协议再写一个回车换行
        outStream.write("/r/n".getBytes());
        //把所有文本类型的实体数据发送出来
        outStream.write(textEntity.toString().getBytes());

        //把所有文件类型的实体数据发送出来
        for(FormFile uploadFile : files)
        {
            StringBuilder fileEntity = new StringBuilder();
            fileEntity.append("--");
            fileEntity.append(BOUNDARY);
            fileEntity.append("/r/n");
            fileEntity.append("Content-Disposition: form-data;name=
            fileEntity.append("Content-Type: "+ uploadFile.getConte
            outStream.write(fileEntity.toString().getBytes());
            //边读边写
            if(uploadFile.getInStream()!=null)
            {
                byte[] buffer = new byte[1024];
                int len = 0;
                while((len = uploadFile.getInStream().read(buffer,
                {
                    outStream.write(buffer, 0, len);
                }
                uploadFile.getInStream().close();
            }
            else
            {
                outStream.write(uploadFile.getData(), 0, uploadFile
            }
            outStream.write("/r/n".getBytes());
        }
        //下面发送数据结束标志，表示数据已经结束
        outStream.write(endline.getBytes());
        BufferedReader reader = new BufferedReader(new InputStreamR
        //读取web服务器返回的数据，判断请求码是否为200，如果不是200，代表请
        if(reader.readLine().indexOf("200")==-1)
        {
            return false;
        }
        outStream.flush();
        outStream.close();
        reader.close();
```

```
        socket.close();
        return true;
    }

    /**
     * 提交数据到服务器
     * @param path 上传路径(注：避免使用localhost或127.0.0.1这样的路径测
     * @param params 请求参数 key为参数名,value为参数值
     * @param file 上传文件
     */
    public static boolean post(String path, Map<String, String> par
    {
       return post(path, params, new FormFile[]{file});
    }
    /**
     * 提交数据到服务器
     * @param path 上传路径(注：避免使用localhost或127.0.0.1这样的路径测
     * @param params 请求参数 key为参数名,value为参数值
     * @param encode 编码
     */
    public static byte[] postFromHttpClient(String path, Map<String
    {
        //用于存放请求参数
        List<NameValuePair> formparams = new ArrayList<NameValuePai
        for(Map.Entry<String, String> entry : params.entrySet())
        {
            formparams.add(new BasicNameValuePair(entry.getKey(), e
        }
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(form
        HttpPost httppost = new HttpPost(path);
        httppost.setEntity(entity);
        //看作是浏览器
        HttpClient httpclient = new DefaultHttpClient();
        //发送post请求
        HttpResponse response = httpclient.execute(httppost);
        return readStream(response.getEntity().getContent());
    }
    /**
     * 发送请求
     * @param path 请求路径
     * @param params 请求参数 key为参数名称 value为参数值
     * @param encode 请求参数的编码
     */
    public static byte[] post(String path, Map<String, String> para
    {
        //String params = "method=save&name="+ URLEncoder.encode("
        StringBuilder parambuilder = new StringBuilder("");
        if(params!=null && !params.isEmpty())
        {
            for(Map.Entry<String, String> entry : params.entrySet()
            {
                parambuilder.append(entry.getKey()).append("=")
                    .append(URLEncoder.encode(entry.getValue(), enc
```

```
            }
            parambuilder.deleteCharAt(parambuilder.length()-1);
        }
        byte[] data = parambuilder.toString().getBytes();
        URL url = new URL(path);
        HttpURLConnection conn = (HttpURLConnection)url.openConnect
        //设置允许对外发送请求参数
        conn.setDoOutput(true);
        //设置不进行缓存
        conn.setUseCaches(false);
        conn.setConnectTimeout(5 * 1000);
        conn.setRequestMethod("POST");
        //下面设置http请求头
        conn.setRequestProperty("Accept", "image/gif, image/jpeg, i
        conn.setRequestProperty("Accept-Language", "zh-CN");
        conn.setRequestProperty("User-Agent", "Mozilla/4.0 (compati
        conn.setRequestProperty("Content-Type", "application/x-www-
        conn.setRequestProperty("Content-Length", String.valueOf(da
        conn.setRequestProperty("Connection", "Keep-Alive");

        //发送参数
        DataOutputStream outStream = new DataOutputStream(conn.getO
        outStream.write(data);//把参数发送出去
        outStream.flush();
        outStream.close();
        if(conn.getResponseCode()==200)
        {
            return readStream(conn.getInputStream());
        }
        return null;
    }

    /**
     * 读取流
     * @param inStream
     * @return 字节数组
     * @throws Exception
     */
    public static byte[] readStream(InputStream inStream) throws Ex
    {
        ByteArrayOutputStream outSteam = new ByteArrayOutputStream(
        byte[] buffer = new byte[1024];
        int len = -1;
        while( (len=inStream.read(buffer)) != -1)
        {
            outSteam.write(buffer, 0, len);
        }
        outSteam.close();
        inStream.close();
        return outSteam.toByteArray();
    }
}
```



偶然发现一篇以前转载的，可以搭配着上面的看看...：使用HttpConnection上传mp3文件

本节小结：

> 本节还是直接无视吧...关于文件上传等进阶部分直接教大家用第三方算了，项目中需要用到 第三方直接复制1的代码，导入个android-async-http即可！

7.3.2 Android 文件下载（1）

本节引言：

> 又是一个深坑，初学者慎入...本节将从普通的单线程下载 -> 普通多线程下载 -> -> 以及一个很实用的例子：利用Android那只DownloadManager更新apk 并覆盖安装的实现代码！好的，这样看上去，本节还是蛮有趣的，开始本节内容！ PS:我们把整个完整的多线程断点续传放到下一节中！

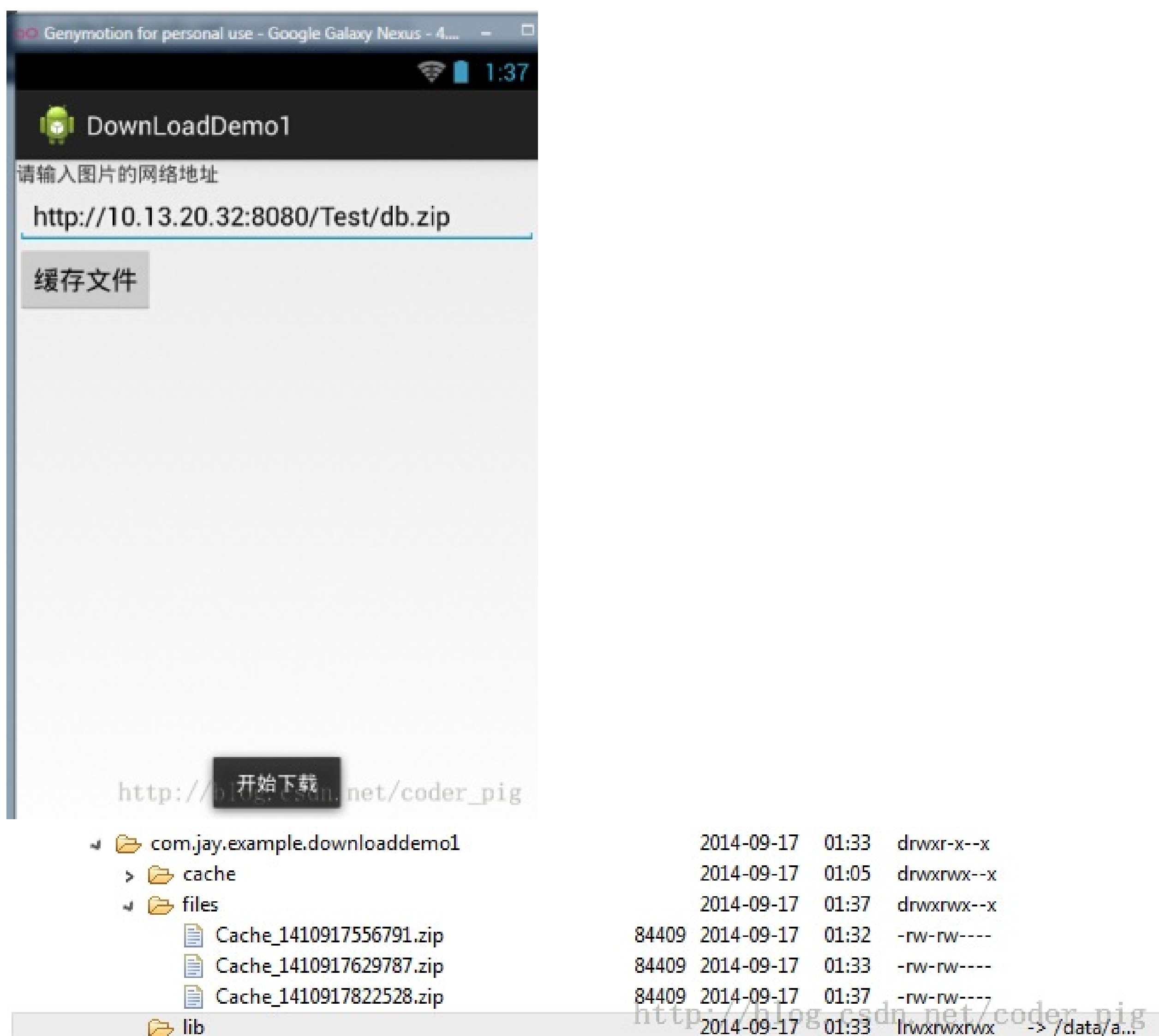
1.普通单线程下载文件：

> 直接使用URLConnection.openStream()打开网络输入流,然后将流写入到文件中！

核心方法：

```
public static void downLoad(String path,Context context)throws Exce
{
  URL url = new URL(path);
  InputStream is = url.openStream();
  //截取最后的文件名
  String end = path.substring(path.lastIndexOf("."));
  //打开手机对应的输出流,输出到文件中
  OutputStream os = context.openFileOutput("Cache_"+System.current
  byte[] buffer = new byte[1024];
  int len = 0;
  //从输入六中读取数据,读到缓冲区中
  while((len = is.read(buffer)) > 0)
  {
    os.write(buffer,0,len);
  }
  //关闭输入输出流
  is.close();
  os.close();
}
```

运行结果：



2.普通多线程下载：

我们都知道使用多线程下载文件可以更快地完成文件的下载,但是为什么呢?

> 答：因为抢占的服务器资源多,假设服务器最多服务100个用户,服务器中的一个线程 对应一个用户100条线程在计算机中并发执行,由CPU划分时间片轮流执行,加入a有99条线程 下载文件,那么相当于占用了99个用户资源,自然就有用较快的下载速度

PS:当然不是线程越多就越好,开启过多线程的话,app需要维护和同步每条线程的开销, 这些开销反而会导致下载速度的降低,另外还和你的网速有关!

多线程下载的流程：

> - 获取网络连接
> - 本地磁盘创建相同大小的空文件
> - 计算每条线程需从文件哪个部分开始下载，结束
> - 依次创建，启动多条线程来下载网络资源的指定部分

普通多线程下载的流程

ps:RandomAccessFile随机访问类,同时整合了FileoutputStream和FileInputStream,支持从文件的任何字节处读写数据;而File只支持将文件作为整体来处理,不能读写文件

①根据要访问的URL路径去调用openConnection(),得到HttpConnection对象,接着嗲用getContentLength();获得下载的文件长度,最后设置本地文件的长度

int filesize = HttpURLConnection.getContentLength();
RandomAccesssFile file = new RandomAccessFile("xxx.apk","rwd");
file.setLength(filesize);

②根据文件长度以及线程数计算每条线程的下载长度,以及每条线程下载的开始位置

计算公式:加入N条线程下载大小为M个字节的文件:
每个线程下载的数据量: M % N == 0 ? M/N : M/N + 1
比如:大小为10个字节的,开3条线程下载,那么每个线程的下载就为10/3+1 = 4,即三条线程的下载量分别为:4,4,2

下载开始位置的计算:假设线程id分别为0,1,2;
那么开始位置 = id * 下载量
结束位置 = (id + 1)*下载量 - 1
(最后一条线程不用计算结束位置的!)

③保存文件,使用RandomAccessFile类指定从文件的什么位置写入数据

RandomAccessFile threadFile = new RandowAccessFile("xxx.apk","rwd");
threadfile.seek(2097152);

另外,在下载时,怎么指定每条线程开始下载的位置呢?
答:HTTP协议为我们提供了一个Range头,使用下面方法指定从什么位置开始下载:
HttpURLConnection.setRequestProperty("Range", "bytes=2097152-4194303");")

PS:这里直接创建一个Java项目，然后在JUnit里运行指定方法即可,

核心代码如下：

```
public class Downloader {
  //添加@Test标记是表示该方法是Junit测试的方法,就可以直接运行该方法了
    @Test
    public void download() throws Exception
    {
      //设置URL的地址和下载后的文件名
      String filename = "meitu.exe";
      String path = "http://10.13.20.32:8080/Test/XiuXiu_Green.exe'
      URL url = new URL(path);
      HttpURLConnection conn = (HttpURLConnection) url.openConnect:
      conn.setConnectTimeout(5000);
      conn.setRequestMethod("GET");
      //获得需要下载的文件的长度(大小)
      int filelength = conn.getContentLength();
      System.out.println("要下载的文件长度"+filelength);
      //生成一个大小相同的本地文件
      RandomAccessFile file = new RandomAccessFile(filename, "rwd")
      file.setLength(filelength);
      file.close();
      conn.disconnect();
      //设置有多少条线程下载
      int threadsize = 3;
      //计算每个线程下载的量
      int threadlength = filelength % 3 == 0 ? filelength/3:fileler
      for(int i = 0;i < threadsize;i++)
      {
        //设置每条线程从哪个位置开始下载
        int startposition = i * threadlength;
        //从文件的什么位置开始写入数据
```

```
          RandomAccessFile threadfile = new RandomAccessFile(filename
          threadfile.seek(startposition);
          //启动三条线程分别从startposition位置开始下载文件
          new DownLoadThread(i,startposition,threadfile,threadlength,
        }
        int quit = System.in.read();
        while('q' != quit)
        {
          Thread.sleep(2000);
        }
      }

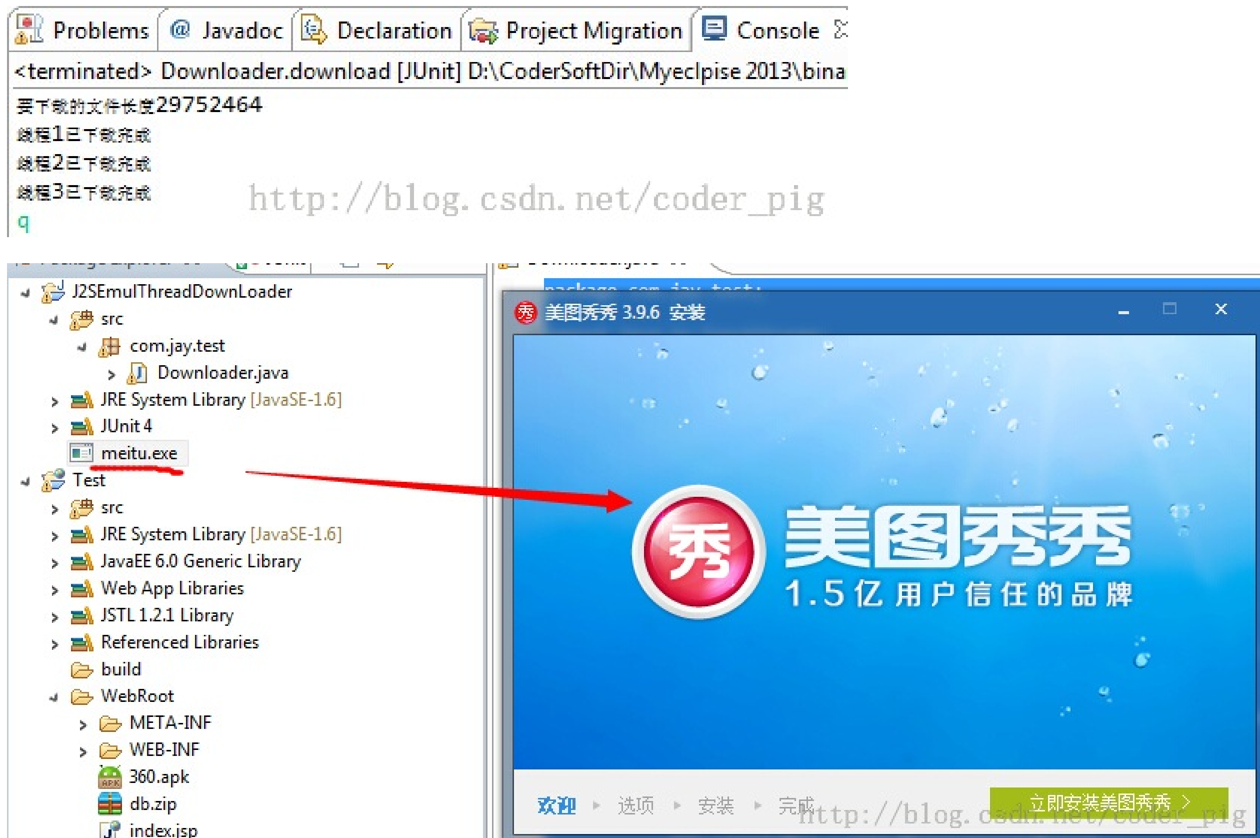
    private class DownLoadThread extends Thread {
      private int threadid;
      private int startposition;
      private RandomAccessFile threadfile;
      private int threadlength;
      private String path;
      public DownLoadThread(int threadid, int startposition,
          RandomAccessFile threadfile, int threadlength, String path)
        this.threadid = threadid;
        this.startposition = startposition;
        this.threadfile = threadfile;
        this.threadlength = threadlength;
        this.path = path;
      }
      public DownLoadThread() {}
      @Override
      public void run() {
        try
        {
          URL url = new URL(path);
          HttpURLConnection conn = (HttpURLConnection) url.openConnec
          conn.setConnectTimeout(5000);
          conn.setRequestMethod("GET");
          //指定从什么位置开始下载
          conn.setRequestProperty("Range", "bytes="+startposition+"-'
          //System.out.println(conn.getResponseCode());
          if(conn.getResponseCode() == 206)
          {
            InputStream is = conn.getInputStream();
            byte[] buffer = new byte[1024];
            int len = -1;
            int length = 0;
            while(length < threadlength && (len = is.read(buffer)) !=
            {
              threadfile.write(buffer,0,len);
              //计算累计下载的长度
              length += len;
            }
            threadfile.close();
            is.close();
            System.out.println("线程"+(threadid+1) + "已下载完成");
```

```
            }
        }catch(Exception ex){System.out.println("线程"+(threadid+1) +
      }

    }
  }
```

运行截图：

如图,使用多线程完成了对文件的下载!双击exe文件可运行,说明文件并没有损坏!



注意事项：

> - int filelength = conn.getContentLength(); //获得下载文件的长度(大小)
> - RandomAccessFile file = new RandomAccessFile(filename, "rwd"); //该类运行对文件进行读写,是多线程下载的核心
> - nt threadlength = filelength % 3 == 0 ? filelength/3:filelength+1; //计算每个线程要下载的量
> - conn.setRequestProperty("Range", "bytes="+startposition+"-"); //指定从哪个位置开始读写,这个是URLConnection提供的方法
> - //System.out.println(conn.getResponseCode()); //这个注释了的代码是用来查看conn的返回码的,我们前面用的都是200, 而针对多线程的话,通常是206,必要时我们可以通过调用该方法查看返回码！
> - int quit = System.in.read();while('q' != quit){Thread.sleep(2000);} //这段代码是做延时操作的,因为我们用的是本地下载,可能该方法运行完了,而我们的 线程还没有开启,这样会引发异常,这里的话,让用户输入一个字符,如果是'q'的话就退出

3.使用DownloadManager更新应用并覆盖安装：

下面的代码可以直接用，加入到项目后，记得为这个内部广播注册一个过滤器：

AndroidManifest.xml

：

```
import android.app.DownloadManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;

/**
 * Created by Jay on 2015/9/9 0009.
 */
public class UpdateAct extends AppCompatActivity {
    //这个更新的APK的版本部分，我们是这样命名的:xxx_v1.0.0_xxxxxxxxx.apk
    //这里我们用的是git提交版本的前九位作为表示
    private static final String FILE_NAME = "ABCDEFGHI";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String endpoint = "";
        try {
            //这部分是获取AndroidManifest.xml里的配置信息的，包名，以及Me
            ApplicationInfo info = getPackageManager().getApplicati
```

```
                getPackageName(), PackageManager.GET_META_DATA)
            //我们在meta_data保存了xxx.xxx这样一个数据，是https开头的一个
            endpoint = info.metaData.getString("xxxx.xxxx").replace
                    "http");
        } catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        }
        //下面的都是拼接apk更新下载url的，path是保存的文件夹路径
        final String _Path = this.getIntent().getStringExtra("path"
        final String _Url = endpoint + _Path;
        final DownloadManager _DownloadManager = (DownloadManager)
        DownloadManager.Request _Request = new DownloadManager.Requ
                Uri.parse(_Url));
        _Request.setDestinationInExternalPublicDir(
                Environment.DIRECTORY_DOWNLOADS, FILE_NAME + ".apk"
        _Request.setTitle(this.getString(R.string.app_name));
        //是否显示下载对话框
        _Request.setShowRunningNotification(true);
        _Request.setMimeType("application/com.trinea.download.file"
        //将下载请求放入队列
        _DownloadManager.enqueue(_Request);
        this.finish();
    }

    //注册一个广播接收器，当下载完毕后会收到一个android.intent.action.DOWN
    //的广播,在这里取出队列里下载任务，进行安装
    public static class Receiver extends BroadcastReceiver {
        public void onReceive(Context context, Intent intent) {
            final DownloadManager _DownloadManager = (DownloadManag
                    .getSystemService(Context.DOWNLOAD_SERVICE);
            final long _DownloadId = intent.getLongExtra(
                    DownloadManager.EXTRA_DOWNLOAD_ID, 0);
            final DownloadManager.Query _Query = new DownloadManage
            _Query.setFilterById(_DownloadId);
            final Cursor _Cursor = _DownloadManager.query(_Query);
            if (_Cursor.moveToFirst()) {
                final int _Status = _Cursor.getInt(_Cursor
                        .getColumnIndexOrThrow(DownloadManager.COLU
                final String _Name = _Cursor.getString(_Cursor
                        .getColumnIndexOrThrow("local_filename"));
                if (_Status == DownloadManager.STATUS_SUCCESSFUL
                        && _Name.indexOf(FILE_NAME) != 0) {

                    Intent _Intent = new Intent(Intent.ACTION_VIEW)
                    _Intent.setDataAndType(
                            Uri.parse(_Cursor.getString(_Cursor
                                    .getColumnIndexOrThrow(Download
                            "application/vnd.android.package-archiv
                    _Intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                    context.startActivity(_Intent);
                }
            }
            _Cursor.close();
```

```
        }
    }
}
```

4.参考代码下载：

普通单线程下载文件：DownLoadDemo1.zip 普通多线程下载文件：J2SEMulDownLoader.zip

本节小结：

> 好的，本节给大家介绍了普通单线程以及多线程下载文件，还有利用Android自带DownManager来 下载更新APK，然后覆盖的实现！相信会对大家的实际开发带来便利，好的，就说这么多，谢谢~

7.3.3 Android 文件下载（2）

本节引言：

> 本节给大家带来的Android中的多线程断点续传的代码解析，呵呵，为什么叫解析呢？因为我 也写不出来，(╯□╰)！先来说说断点的意思吧！所谓的断点就是：使用数据库记录每天线程所 下载的进度！每次启动时根据线程id查询某线程的下载进度，在继续下载！听上去蛮简单的， 要你写十有八九写不出，这很正常，所以本节看懂最好，看不懂也没什么，会用和改就好！ 好的，开始本节内容~

Android多线程断点下载的代码流程解析：

运行效果图：



实现流程全解析：

Step 1：创建一个用来记录线程下载信息的表

创建数据库表,于是乎我们创建一个数据库的管理器类,继承SQLiteOpenHelper类 重写onCreate()与onUpgrade()方法,我们创建的表字段如下:

id	downpath	threadid	downlength

DBOpenHelper.java：

```
package com.jay.example.db;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DBOpenHelper extends SQLiteOpenHelper {
  public DBOpenHelper(Context context) {
    super(context, "downs.db", null, 1);
  }
  @Override
  public void onCreate(SQLiteDatabase db) {
    //数据库的结构为:表名:filedownlog 字段:id,downpath:当前下载的资源,
    //threadid:下载的线程id, downlength:线程下载的最后位置
    db.execSQL("CREATE TABLE IF NOT EXISTS filedownlog " +
        "(id integer primary key autoincrement," +
        " downpath varchar(100)," +
        " threadid INTEGER, downlength INTEGER)");
  }
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newV
    //当版本号发生改变时调用该方法,这里删除数据表,在实际业务中一般是要进行数据
    db.execSQL("DROP TABLE IF EXISTS filedownlog");
    onCreate(db);
  }

}
```

Step 2：创建一个数据库操作类

我们需要创建什么样的方法呢？

> - ①我们需要一个根据URL获得每条线程当前下载长度的方法
> - ②接着,当我们的线程新开辟后,我们需要往数据库中插入与该线程相关参数的方法
> - ③还要定义一个可以实时更新下载文件长度的方法
> - ④我们线程下载完,还需要根据线程id,删除对应记录的方法

FileService.java

```
package com.jay.example.db;

import java.util.HashMap;
import java.util.Map;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

/*
```

```
 * 该类是一个业务bean类,完成数据库的相关操作
 * */

public class FileService {
  //声明数据库管理器
  private DBOpenHelper openHelper;

  //在构造方法中根据上下文对象实例化数据库管理器
  public FileService(Context context) {
    openHelper = new DBOpenHelper(context);
  }

  /**
   * 获得指定URI的每条线程已经下载的文件长度
   * @param path
   * @return
   * */
  public Map<Integer, Integer> getData(String path)
  {
    //获得可读数据库句柄,通常内部实现返回的其实都是可写的数据库句柄
    SQLiteDatabase db = openHelper.getReadableDatabase();
    //根据下载的路径查询所有现场的下载数据,返回的Cursor指向第一条记录之前
    Cursor cursor = db.rawQuery("select threadid, downlength from t
        new String[]{path});
    //建立一个哈希表用于存放每条线程已下载的文件长度
    Map<Integer,Integer> data = new HashMap<Integer, Integer>();
    //从第一条记录开始遍历Cursor对象
    cursor.moveToFirst();
    while(cursor.moveToNext())
    {
      //把线程id与该线程已下载的长度存放到data哈希表中
      data.put(cursor.getInt(0), cursor.getInt(1));
      data.put(cursor.getInt(cursor.getColumnIndexOrThrow("threadic
          cursor.getInt(cursor.getColumnIndexOrThrow("downlength")
    }
    cursor.close();//关闭cursor,释放资源;
    db.close();
    return data;
  }

  /**
   * 保存每条线程已经下载的文件长度
   * @param path 下载的路径
   * @param map 现在的di和已经下载的长度的集合
  */
  public void save(String path,Map<Integer,Integer> map)
  {
    SQLiteDatabase db = openHelper.getWritableDatabase();
    //开启事务,因为此处需要插入多条数据
    db.beginTransaction();
    try{
      //使用增强for循环遍历数据集合
      for(Map.Entry<Integer, Integer> entry : map.entrySet())
```

```
        {
          //插入特定下载路径特定线程ID已经下载的数据
          db.execSQL("insert into filedownlog(downpath, threadid, dow
              new Object[]{path, entry.getKey(), entry.getValue()});
        }
        //设置一个事务成功的标志,如果成功就提交事务,如果没调用该方法的话那么事务
        //就是上面的数据库操作撤销
        db.setTransactionSuccessful();
      }finally{
        //结束一个事务
        db.endTransaction();
      }
      db.close();
    }

    /**
     * 实时更新每条线程已经下载的文件长度
     * @param path
     * @param map
     */
    public void update(String path,int threadId,int pos)
    {
      SQLiteDatabase db = openHelper.getWritableDatabase();
      //更新特定下载路径下特定线程已下载的文件长度
      db.execSQL("update filedownlog set downlength=? where downpath=
          new Object[]{pos, path, threadId});
      db.close();
    }

    /**
     *当文件下载完成后，删除对应的下载记录
     *@param path
     */
    public void delete(String path)
    {
      SQLiteDatabase db = openHelper.getWritableDatabase();
      db.execSQL("delete from filedownlog where downpath=?", new Obje
      db.close();
    }

  }
```

Step 3：创建一个文件下载器类

好了,数据库管理器与操作类都完成了接着就该弄一个文件下载器类了,在该类中又要完成 什么操作呢？要做的事就多了：

> ①定义一堆变量,核心是线程池threads和同步集合ConcurrentHashMap,用于缓存线程下载长度的 ②定义一个获取线程池中线程数的方法; ③定义一个退出下载的方法, ④获取当前文件大小的方法 ⑤累计当前已下载长度的方法,这里需要添加一个synchronized关键字,用来解决并发访问的问题 ⑥更新指定线程最后的下载位置,同样也需要用同步 ⑦在构造方法中完成文件下载,线程开辟等操作 ⑧获取文件名的方法:先截取提供的url最后的'/'后面的字符串,如果获取不到,再从头字段查找,还是 找不到的话,就使用网卡标识数字+cpu的唯一数字生成一个16个字节的二进制作为文件名 ⑨开始下载文件的方法 ⑩获取http响应头字段的方法 ⑪打印http头字段的方法 **12**.打印日志信息的方法

FileDownloadered.java:

```
package com.jay.example.service;

import java.io.File;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.UUID;
import java.util.concurrent.ConcurrentHashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import android.content.Context;
import android.util.Log;

import com.jay.example.db.FileService;

public class FileDownloadered {

  private static final String TAG = "文件下载类";  //设置一个查log时的
  private static final int RESPONSEOK = 200;    //设置响应码为200,代表
  private FileService fileService;        //获取本地数据库的业务Bean
  private boolean exited;             //停止下载的标志
  private Context context;            //程序的上下文对象
  private int downloadedSize = 0;               //已下载的文件长度
  private int fileSize = 0;           //开始的文件长度
  private DownloadThread[] threads;        //根据线程数设置下载的线程池
  private File saveFile;              //数据保存到本地的文件中
  private Map<Integer, Integer> data = new ConcurrentHashMap<Intege
  private int block;                            //每条线程下载的长度
  private String downloadUrl;                   //下载的路径

  /**
   * 获取线程数
   */
  public int getThreadSize()
  {
    //return threads.length;
    return 0;
```

```
    }

    /**
     * 退出下载
     * */
    public void exit()
    {
      this.exited = true;    //将退出的标志设置为true;
    }
    public boolean getExited()
    {
      return this.exited;
    }

    /**
     * 获取文件的大小
     * */
    public int getFileSize()
    {
      return fileSize;
    }

    /**
     * 累计已下载的大小
     * 使用同步锁来解决并发的访问问题
     * */
    protected synchronized void append(int size)
    {
      //把实时下载的长度加入到总的下载长度中
      downloadedSize += size;
    }

    /**
     * 更新指定线程最后下载的位置
     * @param threadId 线程id
     * @param pos 最后下载的位置
     * */
    protected synchronized void update(int threadId,int pos)
    {
      //把指定线程id的线程赋予最新的下载长度,以前的值会被覆盖掉
      this.data.put(threadId, pos);
      //更新数据库中制定线程的下载长度
      this.fileService.update(this.downloadUrl, threadId, pos);
    }

    /**
     * 构建文件下载器
     * @param downloadUrl 下载路径
     * @param fileSaveDir 文件的保存目录
     * @param threadNum  下载线程数
     * @return
     */
    public FileDownloadered(Context context,String downloadUrl,File f
```

```
    {
      try {
        this.context = context;     //获取上下文对象,赋值
        this.downloadUrl = downloadUrl;  //为下载路径赋值
        fileService = new FileService(this.context);   //实例化数据库操
        URL url = new URL(this.downloadUrl);     //根据下载路径实例化URL
        if(!fileSaveDir.exists()) fileSaveDir.mkdir();  //如果文件不存在
        this.threads = new DownloadThread[threadNum];   //根据下载的线程

        HttpURLConnection conn = (HttpURLConnection) url.openConnecti
        conn.setConnectTimeout(5000);      //设置连接超时事件为5秒
        conn.setRequestMethod("GET");      //设置请求方式为GET
        //设置用户端可以接收的媒体类型
        conn.setRequestProperty("Accept", "image/gif, image/jpeg, ima
            "image/pjpeg, application/x-shockwave-flash, application/
            "application/vnd.ms-xpsdocument, application/x-ms-xbap,"
            " application/x-ms-application, application/vnd.ms-excel,
            " application/vnd.ms-powerpoint, application/msword, */*"

        conn.setRequestProperty("Accept-Language", "zh-CN");  //设置用
        conn.setRequestProperty("Referer", downloadUrl);    //设置请求
        conn.setRequestProperty("Charset", "UTF-8");    //设置客户端编
        //设置用户代理
        conn.setRequestProperty("User-Agent", "Mozilla/4.0 (compatibl
            "Windows NT 5.2; Trident/4.0; .NET CLR 1.1.4322; .NET CLR
            " .NET CLR 3.0.04506.30; .NET CLR 3.0.4506.2152; .NET CLR

        conn.setRequestProperty("Connection", "Keep-Alive");  //设置c
        conn.connect();      //和远程资源建立正在的链接,但尚无返回的数据流
        printResponseHeader(conn);   //打印返回的Http的头字段集合
        //对返回的状态码进行判断,用于检查是否请求成功,返回200时执行下面的代码
        if(conn.getResponseCode() == RESPONSEOK)
        {
          this.fileSize = conn.getContentLength();  //根据响应获得文件大
          if(this.fileSize <= 0)throw new RuntimeException("不知道文件
          String filename = getFileName(conn);      //获取文件名称
          this.saveFile = new File(fileSaveDir,filename);  //根据文件保
          Map<Integer,Integer> logdata = fileService.getData(download
          //如果存在下载记录
          if(logdata.size() > 0)
          {
            //遍历集合中的数据,把每条线程已下载的数据长度放入data中
            for(Map.Entry<Integer, Integer> entry : logdata.entrySet(
            {
              data.put(entry.getKey(), entry.getValue());
            }
          }
          //如果已下载的数据的线程数和现在设置的线程数相同时则计算所有现场已经下
          if(this.data.size() == this.threads.length)
          {
            //遍历每条线程已下载的数据
            for(int i = 0;i < this.threads.length;i++)
            {
```

```
                this.downloadedSize += this.data.get(i+1);
            }
            print("已下载的长度" + this.downloadedSize + "个字节");
          }
          //使用条件运算符求出每个线程需要下载的数据长度
          this.block = (this.fileSize % this.threads.length) == 0?
              this.fileSize / this.threads.length:
                this.fileSize / this.threads.length + 1;
        }else{
          //打印错误信息
          print("服务器响应错误:" + conn.getResponseCode() + conn.getRe
          throw new RuntimeException("服务器反馈出错");
        }

    }catch (Exception e)
    {
      print(e.toString());   //打印错误
      throw new RuntimeException("无法连接URL");
    }
  }

  /**
   * 获取文件名
   * */
  private String getFileName(HttpURLConnection conn)
  {
    //从下载的路径的字符串中获取文件的名称
    String filename = this.downloadUrl.substring(this.downloadUrl.l
    if(filename == null || "".equals(filename.trim())){     //如果获
    for(int i = 0;;i++)  //使用无限循环遍历
    {
      String mine = conn.getHeaderField(i);     //从返回的流中获取特定
      if (mine == null) break;            //如果遍历到了返回头末尾则退出循
      //获取content-disposition返回字段,里面可能包含文件名
      if("content-disposition".equals(conn.getHeaderFieldKey(i).toL
        //使用正则表达式查询文件名
        Matcher m = Pattern.compile(".*filename=(.*)").matcher(mine
        if(m.find()) return m.group(1);    //如果有符合正则表达式规则的
      }
    }
    filename = UUID.randomUUID()+ ".tmp";//如果都没找到的话,默认取一个文
    //由网卡标识数字(每个网卡都有唯一的标识号)以及CPU时间的唯一数字生成的一个1
    }
      return filename;
  }

  /**
   *  开始下载文件
   * @param listener 监听下载数量的变化,如果不需要了解实时下载的数量,可以设
   * @return 已下载文件大小
   * @throws Exception
   */
  //进行下载,如果有异常的话,抛出异常给调用者
```

```
  public int download(DownloadProgressListener listener) throws Exc
    try {
      RandomAccessFile randOut = new RandomAccessFile(this.saveFile
      //设置文件大小
      if(this.fileSize>0) randOut.setLength(this.fileSize);
      randOut.close();    //关闭该文件,使设置生效
      URL url = new URL(this.downloadUrl);
      if(this.data.size() != this.threads.length){
      //如果原先未曾下载或者原先的下载线程数与现在的线程数不一致
        this.data.clear();
        //遍历线程池
        for (int i = 0; i < this.threads.length; i++) {
          this.data.put(i+1, 0);//初始化每条线程已经下载的数据长度为0
        }
        this.downloadedSize = 0;   //设置已经下载的长度为0
      }

      for (int i = 0; i < this.threads.length; i++) {//开启线程进行下
        int downLength = this.data.get(i+1);
        //通过特定的线程id获取该线程已经下载的数据长度
        //判断线程是否已经完成下载,否则继续下载
        if(downLength < this.block && this.downloadedSize<this.file
          //初始化特定id的线程
          this.threads[i] = new DownloadThread(this, url, this.save
          //设置线程优先级,Thread.NORM_PRIORITY = 5;
          //Thread.MIN_PRIORITY = 1;Thread.MAX_PRIORITY = 10,数值越
          this.threads[i].setPriority(7);
          this.threads[i].start();    //启动线程
        }else{
          this.threads[i] = null;   //表明线程已完成下载任务
        }
      }

      fileService.delete(this.downloadUrl);
      //如果存在下载记录，删除它们，然后重新添加
      fileService.save(this.downloadUrl, this.data);
      //把下载的实时数据写入数据库中
      boolean notFinish = true;
      //下载未完成
      while (notFinish) {
      // 循环判断所有线程是否完成下载
        Thread.sleep(900);
        notFinish = false;
        //假定全部线程下载完成
        for (int i = 0; i < this.threads.length; i++){
          if (this.threads[i] != null && !this.threads[i].isFinish(
          //如果发现线程未完成下载
            notFinish = true;
            //设置标志为下载没有完成
            if(this.threads[i].getDownLength() == -1){
            //如果下载失败,再重新在已下载的数据长度的基础上下载
            //重新开辟下载线程,设置线程的优先级
              this.threads[i] = new DownloadThread(this, url, this.
```

```
                this.threads[i].setPriority(7);
                this.threads[i].start();
              }
            }
          }
          if(listener!=null) listener.onDownloadSize(this.downloadedS
          //通知目前已经下载完成的数据长度
        }
        if(downloadedSize == this.fileSize) fileService.delete(this.d
        //下载完成删除记录
      } catch (Exception e) {
        print(e.toString());
        throw new Exception("文件下载异常");
      }
      return this.downloadedSize;
    }

    /**
     * 获取Http响应头字段
     * @param http
     * @return
     */
    public static Map<String, String> getHttpResponseHeader(HttpURLCo
      //使用LinkedHashMap保证写入和便利的时候的顺序相同,而且允许空值
      Map<String, String> header = new LinkedHashMap<String, String>(
      //此处使用无线循环,因为不知道头字段的数量
      for (int i = 0;; i++) {
        String mine = http.getHeaderField(i);  //获取第i个头字段的值
        if (mine == null) break;      //没值说明头字段已经循环完毕了,使用b
        header.put(http.getHeaderFieldKey(i), mine); //获得第i个头字段的
      }
      return header;
    }
    /**
     * 打印Http头字段
     * @param http
     */
    public static void printResponseHeader(HttpURLConnection http){
      //获取http响应的头字段
      Map<String, String> header = getHttpResponseHeader(http);
      //使用增强for循环遍历取得头字段的值,此时遍历的循环顺序与输入树勋相同
      for(Map.Entry<String, String> entry : header.entrySet()){
        //当有键的时候则获取值,如果没有则为空字符串
        String key = entry.getKey()!=null ? entry.getKey()+ ":" : "";
        print(key+ entry.getValue());      //打印键和值得组合
      }
    }

    /**
     * 打印信息
     * @param msg 信息字符串
     * */
    private static void print(String msg) {
```

```
        Log.i(TAG, msg);
    }
}
```

Step 4：自定义一个下载线程类

这个自定义的线程类要做的事情如下：

- ① 首先肯定是要继承Thread类啦,然后重写Run()方法
- ② Run()方法:先判断是否下载完成,没有得话:打开URLConnection链接,接着RandomAccessFile 进行数据读写,完成时设置完成标记为true,发生异常的话设置长度为-1,打印异常信息
- ③打印log信息的方法
- ④判断下载是否完成的方法(根据完成标记)
- ⑤获得已下载的内容大小

DownLoadThread.java：

```
package com.jay.example.service;

import java.io.File;
import java.io.InputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URL;

import android.util.Log;

public class DownloadThread extends Thread {
    private static final String TAG = "下载线程类";    //定义TAG,在打印l
    private File saveFile;                  //下载的数据保存到的文件
    private URL downUrl;                  //下载的URL
    private int block;                    //每条线程下载的大小
    private int threadId = -1;              //初始化线程id设置
    private int downLength;               //该线程已下载的数据长度
    private boolean finish = false;          //该线程是否完成下载的标志
    private FileDownloadered downloader;      //文件下载器

    public DownloadThread(FileDownloadered downloader, URL downUrl, F
        this.downUrl = downUrl;
        this.saveFile = saveFile;
        this.block = block;
        this.downloader = downloader;
        this.threadId = threadId;
        this.downLength = downLength;
    }

    @Override
    public void run() {
```

```
    if(downLength < block){//未下载完成
      try {
        HttpURLConnection http = (HttpURLConnection) downUrl.openCo
        http.setConnectTimeout(5 * 1000);
        http.setRequestMethod("GET");
        http.setRequestProperty("Accept", "image/gif, image/jpeg, i
        http.setRequestProperty("Accept-Language", "zh-CN");
        http.setRequestProperty("Referer", downUrl.toString());
        http.setRequestProperty("Charset", "UTF-8");
        int startPos = block * (threadId - 1) + downLength;//开始位置
        int endPos = block * threadId -1;//结束位置
        http.setRequestProperty("Range", "bytes=" + startPos + "-"+
        http.setRequestProperty("User-Agent", "Mozilla/4.0 (compati
        http.setRequestProperty("Connection", "Keep-Alive");

        InputStream inStream = http.getInputStream();     //获得远程
        byte[] buffer = new byte[1024];           //设置本地数据的缓存
        int offset = 0;                       //每次读取的数据量
        print("Thread " + this.threadId + " start download from pos

        RandomAccessFile threadfile = new RandomAccessFile(this.sav
        threadfile.seek(startPos);
        //用户没有要求停止下载,同时没有达到请求数据的末尾时会一直循环读取数据
        while (!downloader.getExited() && (offset = inStream.read(b
          threadfile.write(buffer, 0, offset);          //直接把数据
          downLength += offset;             //把新线程已经写到文件中的数
          downloader.update(this.threadId, downLength); //把该线程已
          downloader.append(offset);             //把新下载的数据长度加
        }
        threadfile.close();
        inStream.close();
        print("Thread " + this.threadId + " download finish");
        this.finish = true;                               //设置完成
      } catch (Exception e) {
        this.downLength = -1;               //设置该线程已经下载的长度为
        print("Thread "+ this.threadId+ ":"+ e);
      }
    }
  }
  private static void print(String msg){
    Log.i(TAG, msg);
  }
  /**
   * 下载是否完成
   * @return
   */
  public boolean isFinish() {
    return finish;
  }
  /**
   * 已经下载的内容大小
   * @return 如果返回值为-1,代表下载失败
   */
```

```
  public long getDownLength() {
    return downLength;
  }
}
```

Step 5：创建一个DownloadProgressListener接口监听下载进度

FileDownloader中使用了DownloadProgressListener进行进度监听, 所以这里需要创建一个接口,同时定义一个方法的空实现:

DownloadProgressListener.java:

```
package com.jay.example.service;
public interface DownloadProgressListener {
  public void onDownloadSize(int downloadedSize);
}
```

Step 6：编写我们的布局代码

另外调用android:enabled="false"设置组件是否可点击, 代码如下

activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.jay.example.multhreadcontinuabledemo.MainAct

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入要下载的文件地址" />
  <EditText
      android:id="@+id/editpath"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="http://10.13.20.32:8080/Test/twelve.mp3"
  />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/btndown"
    android:text="下载"
  />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/btnstop"
    android:text="停止"
    android:enabled="false"
  />

  <ProgressBar
     android:layout_width="fill_parent"
     android:layout_height="18dp"
     style="?android:attr/progressBarStyleHorizontal"
     android:id="@+id/progressBar"
    />

   <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:id="@+id/textresult"
    android:text="显示实时下载的百分比"
    />

</LinearLayout>
```

Step 7：MainActivity的编写

最后就是我们的MainActivity了,完成组件以及相关变量的初始化; 使用handler来完成界面的更新操作,另外耗时操作不能够在主线程中进行, 所以这里需要开辟新的线程,这里用Runnable实现,详情见代码 吧

MainActivity.java:

```
package com.jay.example.multhreadcontinuabledemo;

import java.io.File;

import com.jay.example.service.FileDownloadered;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {

  private EditText editpath;
  private Button btndown;
  private Button btnstop;
  private TextView textresult;
  private ProgressBar progressbar;
  private static final int PROCESSING = 1;   //正在下载实时数据传输Mes
  private static final int FAILURE = -1;     //下载失败时的Message标示

  private Handler handler = new UIHander();

    private final class UIHander extends Handler{
    public void handleMessage(Message msg) {
      switch (msg.what) {
      //下载时
      case PROCESSING:
        int size = msg.getData().getInt("size");     //从消息中获取已
        progressbar.setProgress(size);         //设置进度条的进度
        //计算已经下载的百分比,此处需要转换为浮点数计算
        float num = (float)progressbar.getProgress() / (float)progr
        int result = (int)(num * 100);     //把获取的浮点数计算结果转换
        textresult.setText(result+ "%");   //把下载的百分比显示到界面控
        if(progressbar.getProgress() == progressbar.getMax()){ //下
          Toast.makeText(getApplicationContext(), "文件下载成功", 1)
        }
        break;
```

```
        case FAILURE:    //下载失败时提示
          Toast.makeText(getApplicationContext(), "文件下载失败", 1).sh
          break;
        }
      }
      }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      editpath = (EditText) findViewById(R.id.editpath);
      btndown = (Button) findViewById(R.id.btndown);
      btnstop = (Button) findViewById(R.id.btnstop);
      textresult = (TextView) findViewById(R.id.textresult);
      progressbar = (ProgressBar) findViewById(R.id.progressBar);
      ButtonClickListener listener = new ButtonClickListener();
      btndown.setOnClickListener(listener);
      btnstop.setOnClickListener(listener);

    }

    private final class ButtonClickListener implements View.OnClickLi
      public void onClick(View v) {
        switch (v.getId()) {
        case R.id.btndown:
          String path = editpath.getText().toString();
          if(Environment.getExternalStorageState().equals(Environment
            File saveDir = Environment.getExternalStorageDirectory();
            download(path, saveDir);
          }else{
            Toast.makeText(getApplicationContext(), "sd卡读取失败", 1)
          }
          btndown.setEnabled(false);
          btnstop.setEnabled(true);
          break;

        case R.id.btnstop:
          exit();
          btndown.setEnabled(true);
          btnstop.setEnabled(false);
          break;
        }
      }
      /*
      由于用户的输入事件(点击button, 触摸屏幕....)是由主线程负责处理的，如果主
      此时用户产生的输入事件如果没能在5秒内得到处理，系统就会报“应用无响应”错误。
      所以在主线程里不能执行一件比较耗时的工作，否则会因主线程阻塞而无法处理用户的
      导致“应用无响应”错误的出现。耗时的工作应该在子线程里执行。
       */
      private DownloadTask task;
      /**
```

```
     * 退出下载
     */
    public void exit(){
      if(task!=null) task.exit();
    }
    private void download(String path, File saveDir) {//运行在主线程
      task = new DownloadTask(path, saveDir);
      new Thread(task).start();
    }

    /*
     * UI控件画面的重绘(更新)是由主线程负责处理的，如果在子线程中更新UI控件的
     * 一定要在主线程里更新UI控件的值，这样才能在屏幕上显示出来，不能在子线程
     */
    private final class DownloadTask implements Runnable{
      private String path;
      private File saveDir;
      private FileDownloadered loader;
      public DownloadTask(String path, File saveDir) {
        this.path = path;
        this.saveDir = saveDir;
      }
      /**
       * 退出下载
       */
      public void exit(){
        if(loader!=null) loader.exit();
      }

      public void run() {
        try {
          loader = new FileDownloadered(getApplicationContext(), pa
          progressbar.setMax(loader.getFileSize());//设置进度条的最大
          loader.download(new com.jay.example.service.DownloadProgr
            public void onDownloadSize(int size) {
              Message msg = new Message();
              msg.what = 1;
              msg.getData().putInt("size", size);
              handler.sendMessage(msg);
            }
          });
        } catch (Exception e) {
          e.printStackTrace();
          handler.sendMessage(handler.obtainMessage(-1));
        }
      }
    }
    }
}
```

Step 8：AndroidManifest.xml文件中添加相关权限

```
<!-- 访问internet权限 -->
<uses-permission android:name="android.permission.INTERNET"/>
<!-- 在SDCard中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FIL
<!-- 往SDCard写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_ST
```

参考代码下载：

多线程断点下载器demo：MulThreadContinuableDemo.zip

多线程断点下载+在线音乐播放器：多线程断点下载+在线音乐播放器.zip

本节小结：

> 好的，本节关于Android多线程断点下载的代码解析就这么多，够呛的是把，不过还是 那句话，有别人造好的轮子，为什么还要自己造呢？况且现在的我们还能力造出来， 不是么，So， 暂时弄懂，会用，知道怎么改就好~嗯，就说这么多， 谢谢~

7.4 Android 调用 WebService

本节引言：

> 经过前面的学习，数据请求，数据解析，文件上传下载等，应该满足大家与服务器交互的基本 需求了，而本节给大家介绍的Android调用WebService，其实这玩意有点类似于一些给我们提供 原始数据API服务的数据平台，比如聚合数据！而WebService则用到了XML和SOAP，通过HTTP协议 即可完成与远程机器的交互！嗯，不多说，开始本节内容~

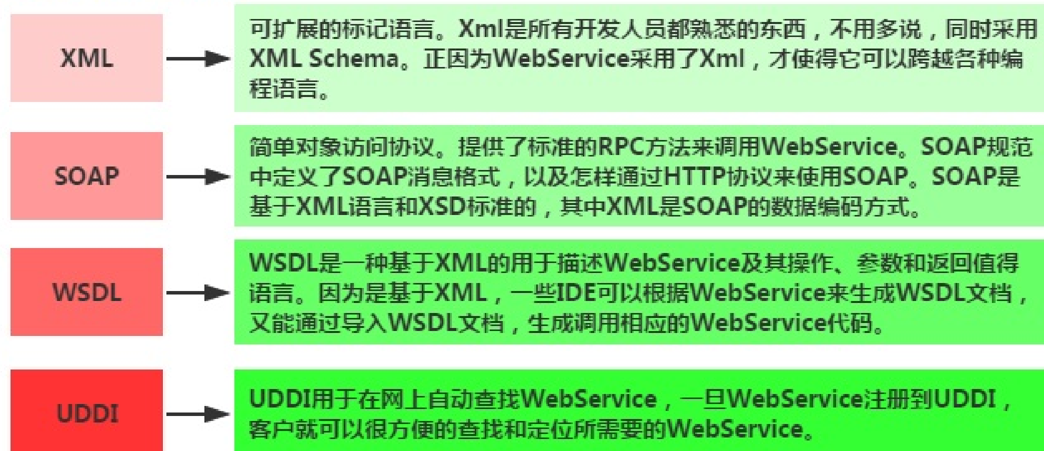
1.WebService简介

Android平台调用WebService

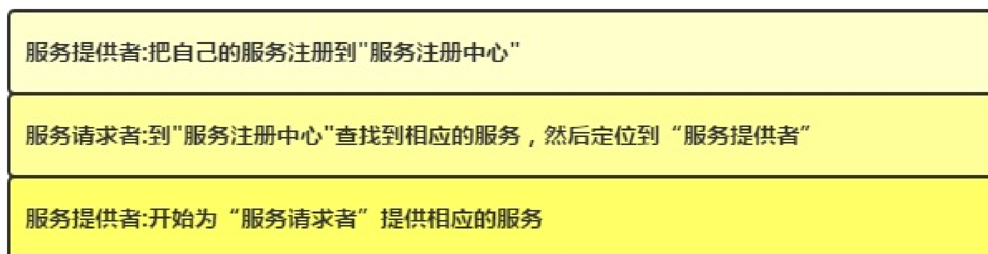
①WebService的引入

我们都知道计手机硬件资源是有限的,对于一些复杂的数据处理,计算,通常都是部署在远程服务器上的,
然后安卓手机作为对应的客户端;为了让android app与远程服务器进行交互,肯定是需要借助一些技术
的了,各种高大上的技术,什么RMI,CORBA,一个都不会,没关系,我们现在会用Web Service平台就够了！
那么.什么是WebService?
简单的说就是某些站点开放出来的服务,当然你也可以自己开发一个service,也就是一些方法，通过URL，指
定某一个方法名，发出请求，站点的这个服务(方法)，接收请求后，根据传入的参数做一些处理，然后将处理
后的结果以XML形式返回给你，你的程序就解析这些XML参数，然后显示出来或做其他操作。
例如：很多大的站点提供有天气预报的webservice、查询某网站的数据的webservice，只要你发送请求过
来，它就返回天气预报、某网站的数据，然后你把结果显示处来。

②主要采用的四个技术



③WebService的模型



PS：如果看完上面简介还不是很清楚的话，那么就算了，之前公司就用C#搭的一个WebService！本节我们并不讨论如何去搭建一个WebService，我们仅仅知道如何去获取WebService提供的服务，然后解析返回的XML数据，然后把相关数据显示到我们的Android设备上就好！

2.去哪里获取WebService服务

网上有很多提供WebService的站点，首先找到这些站点，然后获取相应的服务即可！这里选取WebXml和云聚36wu作为例子给大家讲解下，他们的官网：

webXml：http://www.webxml.com.cn/zh_cn/index.aspx

> 以前是免费的，不过都商业化了，很多服务都要收费，但是可以试用~ 改站点上提供了16个不同的Web服务，可以根据自己的需求，查询相应服务，调用不同的接口！

webXml的相关页面：

相关使用次数说明:

Email电子邮件地址验证WEB服务

接口名称	免费用户	商业用户	说明
ValidateEmailAddress			24小时内不超过50次
ValidateEmailAddressPro			

IP地址来源搜索WEB服务

接口名称	免费用户	商业用户	说明
getCountryCityByIp			免费用户24小时内不超过200次
getGeoIPContext			

中国股票行情数据WEB服务

接口名称	免费用户	商业用户	说明
getStockImageByCode			免费用户24小时内不超过250次；数据延时0-30秒，为商业用户添加图片LOGO水印
getStockImageByteByCode			同上
getStockImage_kByCode			同上
getStockImage_kByteByCode			同上
getStockInfoByCode			免费用户24小时内不超过250次；数据延时0-30秒

中国股票行情分时走势预览缩略图WEB服务

接口名称	免费用户	商业用户	说明
getSmallImage			免费用户24小时内不超过250次
getSmallImageByte			同上

中国邮政编码 <-> 地址信息双向查询/搜索WEB服务

接口名称	免费用户	商业用户	说明
getAddressByZipCode			免费用户24小时内不超过60次

云聚**36wu**：http://www.36wu.com/Service

> 同样也提供了很多的服务,很多手机的app都是用的这里的接口,比如彩虹公交,手机天气等 不过,这个也是要收费的=-=,可以试用,不过只能一小时内发送20次请求; 点击申请使用,获得key就可以了!两者随便选一个吧!

首页 数据与接口 经典案例 支持服务 文档说明 关于我们 申请试用

数据与接口　　累计调用次数：24,211,12

全国天气查询 快递/物流跟踪 火车票/时刻查询 翻译 IP地址查询 手机归属地 全国公交查询 全国车辆违章查询

身份证查询 ISBN信息查询 全国POI信息 影片/影讯查询

3.第三方jar包的准备

> 首先如果想在Android平台上调用WebService需要依赖于第三方类库:ksoap2 而在Android平台上,使用的是ksoap2 Android,一个高效,轻量级的SOAP开发包!
>
> jar包下载地址：https://code.google.com/p/ksoap2-android/wiki/HowToUse?tm=2
>
> 天朝可能上不去，这里提供两个百度云的链接供大家下载使用：
>
> **2.54**版本：ksoap2-android 2.54.jar
>
> **3.30**版本：ksoap2-android 3.30.jar
>
> 如果所幸你能进入jar包的下载地址的话,那么你会看到下面的界面:

---

ksoap2-android
A lightweight and efficient SOAP library for the Android platform.

Project Home | Downloads | Wiki | Issues | Source

General

- How to use
- Project News
- ProjectsUsingKsoap2Android
- Coding Tips and Tricks
- Cookies
- You need help?
- Useful links
- History
- Why contribute?
- Mailing List
- License Information

Contributors

- Obtaining the source
- How to contribute
- Working with the source code

HowToUse
Find out how to use this library in your Android application

How to Use

Apache Maven

To make use of this library in a Maven project, add the following to your project's POM:

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>com.google.code.ksoap2-android</groupId>
      <artifactId>ksoap2-android</artifactId>
      <version>3.3.0</version>
    </dependency>
  </dependencies>
  ...
</project>
```

and configure a proxy repository in your repository manager using the url

```
http://ksoap2-android.googlecode.com/svn/m2-repo
```

If you are not using a repository manager like Sonatype Nexus, you should start doing so now.

As a stop gap solution you could add the repository to a profile in your settings.xml or pom.xml:

```
...
<repositories>
  <repository>
```

http://blog.csdn.net/coder_pig

```
<repositories>
  <repository>
    <id>googlecode-ksoap2-android</id>
    <name>googlecode-ksoap2-android</name>
    <url>http://ksoap2-android.googlecode.com/svn/m2-repo</url>
  </repository>
</repositories>
```

Contact simpligility technologies for consulting and training for Sonatype Nexus.

Further libraries as part of the project are also published as part of the build and can be declared as dependencies as well, if needed.

Standard Android Ant-based build and Eclipse/ADT

To make use of this library in a non-Maven project, follow the instructions in the Android Developer's Guide on how to Add an External Library to your project.

You will need to add a ksoap2-android and all required transitive dependencies to the build path. Luckily the Maven build of the project produces a nice bundle of all these jars in one big file.

The different version of these files are available at http://code.google.com/p/ksoap2-android/source/browse/#svn/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/

To download a file from there, right click on "View raw file" and select "Save Link as" (this label differs for different browsers) and you will get the full jar downloaded.

随便点击一个下载就可以了,如果下载不了;
可以使用迅雷或者QQ旋风进行下载
当前最新版本为3.3.0

The latest release artifact would be available at

http://code.google.com/p/ksoap2-android/source/browse/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/3.3.0/ksoap2-android-assembly-3.3.0-jar-with-dependencies.jar

with a direct download url of

http://ksoap2-android.googlecode.com/svn/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/3.3.0/ksoap2-android-assembly-3.3.0-jar-with-dependencies.jar

Some users seem to experience download problems with IE. Just try a decent browser or download with a command line tool like wget

```
wget http://ksoap2-android.googlecode.com/svn/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/3.3.0/ks
```

After the manual download confirm the size of the downloaded file since many users seem to be experiencing problems.

4.获取相关的一些参数

首先找到我们需要获取的服务，然后记录相关的参数： **NameSpace**(命名空间),**SoapAction**以及**URL**就不用说了,其他参数这样找:

国内手机号码归属地查询WEB服务 通讯和通信
Endpoint: http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx
Disco: http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx?disco
WSDL: http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl

国内手机号码归属地查询WEB服务，提供最新的国内手机号码段归属地数据，每月更新。包括最新的电信天翼189号段和最新移动152号段、TD-SCDMA188号段。数据更全更准确，是目前国内最新最全的手机号码段数据库！

P.1

2500多个城市天气预报 WEB服务 公用事业
Endpoint: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx
Disco: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx?disco
WSDL: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl

2500多个城市天气预报Web服务，包含2400个以上中国城市和100个以上国外城市天气预报数据。数据每2.5小时左右自动更新一次，准确可靠。为让更多的开发人员学习WEB服务开发，此服务支持免费用户使用。为支持多种平台开发，此WEB服务接口提供了多种返回类型可选择。

比如我们这里找的是天气的查询参数，点进去我们可以看到这样一个参数文档：

WeatherWS

WebXml.com.cn **2400多个城市天气预报Web服务**，包含2300个以上中国城市和100个以上国外城
使用本站 WEB 服务请注明或链接本站：http://www.webxml.com.cn/ 感谢大家的支持！

接口帮助文档　部分城市介绍和气候背景　部分城市图片　天气现象和图例

支持下列操作。有关正式定义，请查看服务说明。

- getRegionCountry

  获得国外国家名称和与之对应的ID

  输入参数：无，返回数据：一维字符串数组。

- getRegionDataset

  获得中国省份、直辖市、地区；国家名称（国外）和与之对应的ID

  输入参数：无，返回数据：DataSet。

- getRegionProvince

  获得中国省份、直辖市、地区和与之对应的ID

  输入参数：无，返回数据：一维字符串数组。

- getSupportCityDataset

  获得支持的城市/地区名称和与之对应的ID

  输入参数：theRegionCode = 省市、国家ID或名称，返回数据：DataSet。

- getSupportCityString

  获得支持的城市/地区名称和与之对应的ID

  输入参数：theRegionCode = 省市、国家ID或名称，返回数据：一维字符串数组。

- getWeather

  获得天气预报数据

  输入参数：城市/地区ID或名称，返回数据：一维字符串数组。

比如这里我们需要的是天气查询部分的功能：

WeatherWS

单击此处，获取完整的操作列表。

getWeather

获得天气预报数据

输入参数：城市/地区ID或名称，返回数据：一维字符串数组。

测试

若要使用 HTTP POST 协议对操作进行测试，请单击"调用"按钮。

参数	值
theCityCode:	
theUserID:	

调用

SOAP 1.1

以下是 SOAP 1.2 请求和响应示例。所显示的占位符需替换为实际值。

```
POST /WebServices/WeatherWS.asmx HTTP/1.1
Host: webservice.webxml.com.cn
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://WebXml.com.cn/getWeather"      SoapAction

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
  <soap:Body>
    <getWeather xmlns="http://WebXml.com.cn/">      NameSpace(命名空间)
      <theCityCode>string</theCityCode>
      <theUserID>string</theUserID>
    </getWeather>
  </soap:Body>
</soap:Envelope>
```

先把框住的SoapAction和NameSpace拷贝下来！当然我们可以在这个页面测试，另外我们是免费用户，id可以不填直接跳过，输入后点击调用按钮会打开这样一个页面：

This XML file does not appear to have any style informatio


```
▼<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
    <string>广东 珠海</string>
    <string>珠海</string>
    <string>2417</string>
    <string>2015/09/10 10:32:11</string>
    <string>今日天气实况：气温：29℃；风向/风力：东风 1级；湿度：72%</st
    <string>空气质量：暂无；紫外线强度：很强</string>
  ▼<string>
      太阳镜指数：很必要。建议佩戴透射比2级且UV400的遮阳镜。 穿衣指数：
      霜。 感冒指数：少发。感冒机率较低，避免长期处于空调屋中。 空气污染
    </string>
    <string>9月10日 多云</string>
    <string>27℃/31℃</string>
    <string>无持续风向微风</string>
    <string>1.gif</string>
    <string>1.gif</string>
    <string>9月11日 多云</string>
    <string>26℃/31℃</string>
    <string>无持续风向微风</string>
    <string>1.gif</string>
    <string>1.gif</string>
    <string>9月12日 多云</string>
    <string>25℃/31℃</string>
    <string>无持续风向微风转北风3-4级</string>
    <string>1.gif</string>
    <string>1.gif</string>
    <string>9月13日 小雨转多云</string>
    <string>26℃/31℃</string>
    <string>北风3-4级转无持续风向微风</string>
    <string>7.gif</string>
    <string>1.gif</string>
    <string>9月14日 多云</string>
    <string>27℃/32℃</string>
    <string>无持续风向微风</string>
    <string>1.gif</string>
    <string>1.gif</string>
  </ArrayOfString>
```


嘿嘿，这里就是返回的XML，而我们要做的也就是解析这样一个XML，另外这里的 .gif代表的是天气图标！

同理，我们再把归属地查询的看下SoapAction，NameSpace以及相关参数mark下！

MobileCodeWS

单击此处，获取完整的操作列表。

getMobileCodeInfo

获得国内手机号码归属地省份、地区和手机卡类型信息

输入参数：mobileCode = 字符串（手机号码，最少前7位数字），userID = 字符串（商业用户ID） 免费用户为空字符串；讠

测试

若要使用 HTTP POST 协议对操作进行测试，请单击"调用"按钮。

参数	值
mobileCode:	13798983314
userID:	

调用

SOAP 1.1

以下是 SOAP 1.2 请求和响应示例。所显示的占位符需替换为实际值。

```
POST /WebServices/MobileCodeWS.asmx HTTP/1.1
Host: webservice.webxml.com.cn
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://WebXml.com.cn/getMobileCodeInfo"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.c
  <soap:Body>
    <getMobileCodeInfo xmlns="http://WebXml.com.cn/">
      <mobileCode>string</mobileCode>
      <userID>string</userID>
    </getMobileCodeInfo>
  </soap:Body>
</soap:Envelope>
```

以及返回后的**XML**数据：

This XML file does not appear to have any style information associated with it. The

```
<string xmlns="http://WebXml.com.cn/">13798       : 广东 珠海 广东移动动感地带卡</string>
```

5.注册并启用相关WEB服务

新会员注册 接受条款 填写资料

*会员（登录）名

*称呼 ○小姐 ◉先生

*登录密码

*确认密码

*电子邮件 认真填写

*验证码 7KA8

--------- 以上表单内容必须填写，电子邮件需要确认后才能成为会员。以下为选填项

真实姓名

职务

公司名称

联系地址

邮政编码

电话号码 e.g.021-12345678-00

传真号码 e.g.021-12345678-00

移动电话

网站URL

QQ/MSN

☐ 我已阅读、理解并接受 [Ideabody.Inc 会员注册条款]

点击我的Web服务器，然后点击试用，WebXML给我们提供了五天的免费试用，我们把需要的两个服务器开启！

好的，记得mark下我们自己的key哦~

6.调用WebService的代码示例

嗯，接下来我们来写代码验证调用WebService的流程：

运行效果图：



PS:这个号码是以前的号码=-=，别尝试拨打，已经换人了~ 另外天气服务好像有写问题，有时并不能获取到，估计是WebXml做的一些限制， 毕竟试用...

实现代码：

```
public class MainActivity extends AppCompatActivity implements View

    private EditText edit_param;
    private Button btn_attribution;
    private Button btn_weather;
    private TextView txt_result;

    private String city;
    private String number;
    private String result;

    //定义获取手机信息的SoapAction与命名空间,作为常量
    private static final String AddressnameSpace = "http://WebXml.c
    //天气查询相关参数
    private static final String Weatherurl = "http://webservice.web
    private static final String Weathermethod = "getWeather";
    private static final String WeathersoapAction = "http://WebXml.
    //归属地查询相关参数
```

```
    private static final String Addressurl = "http://webservice.wet
    private static final String Addressmethod = "getMobileCodeInfo'
    private static final String AddresssoapAction = "http://WebXml.

    //定义一个Handler用来更新页面：
    private Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case 0x001:
                    txt_result.setText("结果显示：\n" + result);
                    Toast.makeText(MainActivity.this, "获取天气信息成
                    break;
                case 0x002:
                    txt_result.setText("结果显示：\n" + result);
                    Toast.makeText(MainActivity.this, "号码归属地查询
                    break;
            }

        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        edit_param = (EditText) findViewById(R.id.edit_param);
        btn_attribution = (Button) findViewById(R.id.btn_attributic
        btn_weather = (Button) findViewById(R.id.btn_weather);
        txt_result = (TextView) findViewById(R.id.txt_result);
        btn_attribution.setOnClickListener(this);
        btn_weather.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_weather:
                new Thread() {
                    @Override
                    public void run() {
                        getWether();
                    }
                }.start();
                break;
            case R.id.btn_attribution:
                new Thread(new Runnable() {
                    public void run() {
                        getland();
                    }
```

```
                }).start();
                break;
        }
    }

    //定义一个获取某城市天气信息的方法：
    public void getWether() {
        result = "";
        SoapObject soapObject = new SoapObject(AddressnameSpace, We
        soapObject.addProperty("theCityCode:", edit_param.getText()
        soapObject.addProperty("theUserID", "dbdf15804762404587849
        SoapSerializationEnvelope envelope = new SoapSerializationE
        envelope.bodyOut = soapObject;
        envelope.dotNet = true;
        envelope.setOutputSoapObject(soapObject);
        HttpTransportSE httpTransportSE = new HttpTransportSE(Weath
        System.out.println("天气服务设置完毕,准备开启服务");
        try {
            httpTransportSE.call(WeathersoapAction, envelope);
//              System.out.println("调用WebService服务成功");
        } catch (Exception e) {
            e.printStackTrace();
//              System.out.println("调用WebService服务失败");
        }

        //获得服务返回的数据,并且开始解析
        SoapObject object = (SoapObject) envelope.bodyIn;
        System.out.println("获得服务数据");
        result = object.getProperty(1).toString();
        handler.sendEmptyMessage(0x001);
        System.out.println("发送完毕,textview显示天气信息");
    }

    //定义一个获取号码归属地的方法：
    public void getland() {
        result = "";
        SoapObject soapObject = new SoapObject(AddressnameSpace, Ad
        soapObject.addProperty("mobileCode", edit_param.getText().t
        soapObject.addProperty("userid", "dbdf1580476240458784992
        SoapSerializationEnvelope envelope = new SoapSerializationE
        envelope.bodyOut = soapObject;
        envelope.dotNet = true;
        envelope.setOutputSoapObject(soapObject);
        HttpTransportSE httpTransportSE = new HttpTransportSE(Addre
        //    System.out.println("号码信息设置完毕,准备开启服务");
        try {
            httpTransportSE.call(AddresssoapAction, envelope);
            //System.out.println("调用WebService服务成功");
        } catch (Exception e) {
            e.printStackTrace();
            //System.out.println("调用WebService服务失败");
        }
```

```
            //获得服务返回的数据,并且开始解析
            SoapObject object = (SoapObject) envelope.bodyIn;//System.c
            result = object.getProperty(0).toString();//System.out.prir
            handler.sendEmptyMessage(0x001);
            //System.out.println("发送完毕,textview显示天气信息");
        }

    }
```

另外，别忘了导包和Internet的权限！

```
<uses-permission android:name="android.permission.INTERNET"/>
```

参考代码下载：

WebServiceDemo.zip：下载 WebServiceDemo.zip

本节小结：

> 好的，本节关于Android端如何去使用这个WebService就讲解到这里，下一节我们来学习一个 类似于浏览器的Android控件——WebView，敬请期待~谢谢~！

7.5.1 WebView(网页视图)基本用法

本节引言

> 本节给大家带来的是Android中的一个用于显示网页的控件：**WebView**(网页视图)。
>
> 现在Android应用 层开发的方向有两种：客户端开发和HTML5移动端开发！
>
> 所谓的HTML5端就是：HTML5 + CSS + JS来构建 一个网页版的应用,而这中间的媒介就是这个WebView,而Web和网页端可以通过JS来进行交互,比如, 网页读取手机联系人,调用手机相关的API等！
>
> 而且相比起普通的客户端开发,HTML5移动端有个优势： 可以用百分比来布局,而且如果HTML5端有什么大改,我们不用像客户端那样去重新下一个APP,然后覆盖安装,我们只需修改下网页即可！而客户端...惨不忍睹,当然HTML5也有个缺点,就是性能的问题, 数据积累,耗电问题,还有闪屏等等...
>
> 另外,针对这种跨平台我们可以使用其他的第三方快速开发 框架,比如PhoneGap,对了,还有现在网络上很多一键生成APP类的网站,用户通过拖拉,设置图片 之类的简单操作就可以生成一个应用,大部分都是用的HTML5来完成的！有模板,直接套,你懂的~ 好的,话不多说,开始本节内容！

1.什么是WebView？

> 答：Android内置webkit内核的高性能浏览器,而WebView则是在这个基础上进行封装后的一个 控件,WebView直译网页视图,我们可以简单的看作一个可以嵌套到界面上的一个浏览器控件！

2.相关方法

> 先上官方文档：WebView 并不打算一个个地去讲属性,用到哪个写哪个,其他的自行查阅文档！ 除了直接WebView外我们还可以添加你自己的行为,可以自行定制下述类：

WebChromeClient：辅助WebView处理Javascript的对话框、网站图标、网站title、加载进度等！ 比如下面这些：

方法	作用
onJsAlert(WebView view,String url,String message,JsResult result)	处理Js中的Alert对话框
onJsConfirm(WebView view,String url,String message,JsResult result)	处理Js中的Confirm对话框
onJsPrompt(WebView view,String url,String message,String defaultValue,JsPromptResult result)	处理Js中的Prompt对话框
onProgressChanged(WebView view,int newProgress)	当加载进度条发生改变时调用
onReceivedIcon(WebView view, Bitmap icon)	获得网页的icon
onReceivedTitle(WebView view, String title)	获得网页的标题

WebViewClient：辅助WebView处理各种通知与请求事件！ 比如下面这些方法：

方法	作用
onPageStared(WebView view,String url)	通知主程序网页开始加载
onPageFinished(WebView view,String url,Bitmap favicon)	通知主程序,网页加载完毕
doUpdateVisitedHistory(WebView view,String url,boolean isReload)	更新历史记录
onLoadResource(WebView view,String url)	通知主程序WebView即将加载指定url的资源
onScaleChanged(WebView view,float oldScale,float newScale)	ViewView的缩放发生改变时调用
shouldOverrideKeyEvent(WebView view,KeyEvent event)	控制webView是否处理按键时间,如果返回true,则WebView不处理,返回false则处理
shouldOverrideUrlLoading(WebView view,String url)	控制对新加载的Url的处理,返回true,说明主程序处理WebView不做处理,返回false意味着WebView会对其进行处理
onReceivedError(WebView view,int errorCode,String description,String failingUrl)	遇到不可恢复的错误信息时调用

WebSettings：WebView相关配置的设置，比如setJavaScriptEnabled()设置是否允许JS脚本执行 部分方法如下：

方法	作用
getSettings()	返回一个WebSettings对象,用来控制WebView的属性设置
loadUrl(String url)	加载指定的Url
loadData(String data,String mimeType,String encoding)	加载指定的Data到WebView中.使用"data:"作为标记头,该方法不能加载网络数据.其中mimeType为数据类型如:textml,image/jpeg. encoding为字符的编码方式
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl)	比上面的loadData更加强大
setWebViewClient(WebViewClient client)	为WebView指定一个WebViewClient对象.WebViewClient可以辅助WebView处理各种通知,请求等事件。
setWebChromeClient(WebChromeClient client)	为WebView指定一个WebChromeClient对象,WebChromeClient专门用来辅助WebView处理js的对话框,网站title,网站图标,加载进度条等

这里重要区分三个load方法的区别：

> **loadUrl**()：直接显示网页内容(单独显示网络图片)，一般不会出现乱码。
> **loadData**(data, "text/html", "UTF-8")：用来加载URI格式的数据，不能通过网络来加载内容， 不能加载图片，而且经常会遇到乱码的问题，我们知道String类型的数据主要是Unicode编码的， 而WebView一般为了节省资源使用的是UTF-8编码，尽管我们按上面写了，但是还需要为webView设置：webview.getSettings().setDefaultTextEncodingName("UTF -8");
> **loadDataWithBaseURL**(baseUrl, string, "text/html", "utf-8", null)：loadData类的一个 增强类， 可以加载图片，baseUrl为你存储的图片路径，而且只需在这里设置utf-8就可以解决乱码 问题了！

这里只是列举了部分属性而已，其他的还需自行查阅官方文档：

WebChromeClient文档

WebViewClient文档

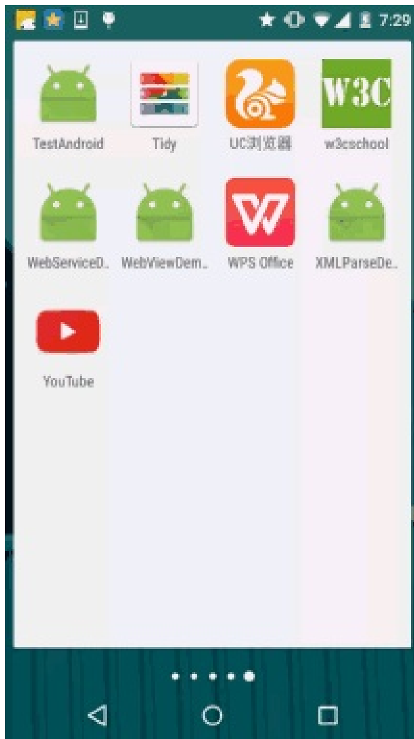
WebSettings文档

3.一些常见需求讲解

需求1：根据URL加载网页

1）直接在Activity上加载一个WebView

运行效果图：



实现代码：

```
public class MainActivity extends AppCompatActivity {

    private WebView webView;
    private long exitTime = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        webView = new WebView(this);
        webView.setWebViewClient(new WebViewClient() {
            //设置在webView点击打开的新网页在当前界面显示,而不跳转到新的浏览
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, S
                view.loadUrl(url);
                return true;
            }
        });
        webView.getSettings().setJavaScriptEnabled(true);  //设置We
        webView.loadUrl("http://www.baidu.com/");          //调用lo
        setContentView(webView);                           //调用Ac
    }

    //我们需要重写回退按钮的时间,当用户点击回退按钮：
    //1.webView.canGoBack()判断网页是否能后退,可以则goback()
    //2.如果不可以连续点击两次退出App,否则弹出提示Toast
    @Override
    public void onBackPressed() {
        if (webView.canGoBack()) {
            webView.goBack();
        } else {
            if ((System.currentTimeMillis() - exitTime) > 2000) {
                Toast.makeText(getApplicationContext(), "再按一次退出
                        Toast.LENGTH_SHORT).show();
                exitTime = System.currentTimeMillis();
            } else {
                super.onBackPressed();
            }

        }
    }
}
```

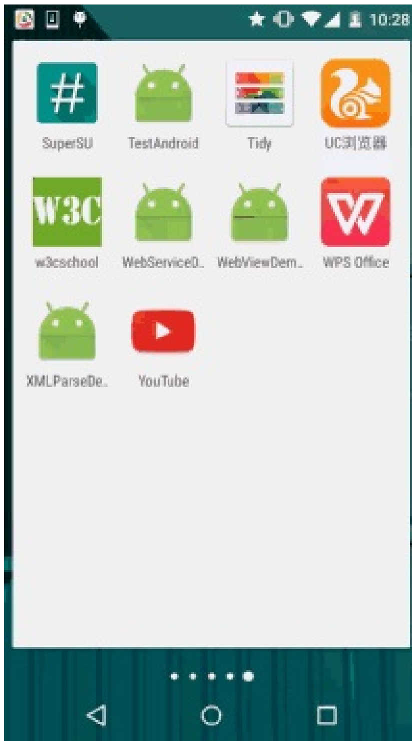
2）布局代码中设置WebView

相信大家都见过很多的新闻类App吧或者门户信息类的App,他的结构可能是这样的：



左上角一个点击关闭当前Activity的按钮,中间是新闻的title,右面是一个刷新按钮, 而在右下角可能有这样一个悬浮的按钮,当我们滑动超过屏幕宽度他就会显示出来, 当用户点击后又会回滚到网页的顶部！下面我们来简单的实现下！

运行效果图：



实现代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_back;
    private TextView txt_title;
    private Button btn_top;
    private Button btn_refresh;
    private WebView wView;
    private long exitTime = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_back = (Button) findViewById(R.id.btn_back);
        txt_title = (TextView) findViewById(R.id.txt_title);
        btn_top = (Button) findViewById(R.id.btn_top);
        btn_refresh = (Button) findViewById(R.id.btn_refresh);
        wView = (WebView) findViewById(R.id.wView);

        btn_back.setOnClickListener(this);
        btn_refresh.setOnClickListener(this);
        btn_top.setOnClickListener(this);
```

```
        wView.loadUrl("http://www.baidu.com");
        wView.setWebChromeClient(new WebChromeClient() {
            //这里设置获取到的网站title
            @Override
            public void onReceivedTitle(WebView view, String title)
                super.onReceivedTitle(view, title);
                txt_title.setText(title);
            }
        });

        wView.setWebViewClient(new WebViewClient() {
            //在webview里打开新链接
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, S
                view.loadUrl(url);
                return true;
            }
        });
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_back:
                finish();          //关闭当前Activity
                break;
            case R.id.btn_refresh:
                wView.reload();    //刷新当前页面
                break;
            case R.id.btn_top:
                wView.setScrollY(0);   //滚动到顶部
                break;
        }
    }

    @Override
    public void onBackPressed() {
        if (wView.canGoBack()) {
            wView.goBack();
        } else {
            if ((System.currentTimeMillis() - exitTime) > 2000) {
                Toast.makeText(getApplicationContext(), "再按一次退出
                        Toast.LENGTH_SHORT).show();
                exitTime = System.currentTimeMillis();
            } else {
                finish();
            }

        }
    }
}
```

问题答疑：

相信细心的朋友看到，我们回到一开始加载的页面后，按返回键，按了多次还是没有退出 当前的APP，后来还是我们手动去点back键通过调用finish方法才能关闭当前的Activity? 这是为什么呢？明明百度一下已经是第一个页面啊？

> 答：其实发生这个的原因是：网址的重定向问题引起的，其实我们在访问百度的时候：
>
> 尽管我们load的是www.baidu.com，但是百度做了重定向，跳转到了手机版百度一下网页： 即实际你的流程是：www.baidu.com -> 手机版百度一下 -> 打开其他的链接！
>
> 我们看到我们上面shouldOverrideUrlLoading()方法是这样写的：
>
> **view.loadUrl(url);return true;** 我们知道用户点击一次回退键，那么webview会调用一次goback方法()，我们把上面三个 设做A,B,C三个站点，在C时点回退，C - > B没问题，接着再点 B -> A，这个时候问题 就来了尽管B来到了A，但是因为重定向又跳转到了B，如此循环往复...这就是为什么 点击回退键并没有推出WebView的原因，解决方法：手速，在webview未加载完网页 钱连续双击回退键，手速要够快，哈哈！说笑而已，要解决这个问题，我们只需将shouldOverrideUrlLoading里的东东删掉，然后写上**return false**；即可！ 不信是重定向，可以自己修改下URL试试~

需求2：WebView滚动事件的监听

我们都知道监听滚动事件一般都是设置setOnScrollChangedListener，可惜的是WebView并没有给我们提供这样的方法，但是我们可以重写WebView，覆盖里面的一个方法： protected void onScrollChanged(final int l, final int t, final int oldl,final int oldt){} 然后再对外提供一个接口，示例代码如下：

MyWebViewDemo.java：

```
/**
 * Created by Jay on 2015/9/11 0011.
 */
public class MyWebView extends WebView {

    private OnScrollChangedCallback mOnScrollChangedCallback;

    public MyWebView(Context context) {
        super(context);
    }

    public MyWebView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public MyWebView(Context context, AttributeSet attrs, int defSt
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onScrollChanged(int l, int t, int oldl, int oldt
        super.onScrollChanged(l, t, oldl, oldt);
        if (mOnScrollChangedCallback != null) {
            mOnScrollChangedCallback.onScroll(l - oldl, t - oldt);
        }
    }

    public OnScrollChangedCallback getOnScrollChangedCallback() {
        return mOnScrollChangedCallback;
    }

    public void setOnScrollChangedCallback(
            final OnScrollChangedCallback onScrollChangedCallback)
        mOnScrollChangedCallback = onScrollChangedCallback;
    }

    public static interface OnScrollChangedCallback {
        //这里的dx和dy代表的是x轴和y轴上的偏移量，你也可以自己把l, t, oldl,
        public void onScroll(int dx, int dy);
    }

}
```

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    private MyWebView wView;
    private Button btn_icon;
    private long exitTime = 0;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn_icon = (Button) findViewById(R.id.btn_icon);
        wView = (MyWebView) findViewById(R.id.wView);
        wView.loadUrl("http://www.hao123.com");
        wView.setWebViewClient(new WebViewClient() {
            //在webview里打开新链接
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, S
                view.loadUrl(url);
                return true;
            }
        });

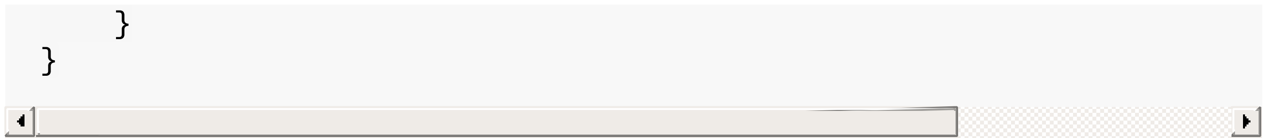
        //比如这里做一个简单的判断，当页面发生滚动，显示那个Button
        wView.setOnScrollChangedCallback(new MyWebView.OnScrollChar
            @Override
            public void onScroll(int dx, int dy) {
                if (dy > 0) {
                    btn_icon.setVisibility(View.VISIBLE);
                } else {
                    btn_icon.setVisibility(View.GONE);
                }
            }
        });

        btn_icon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                wView.setScrollY(0);
                btn_icon.setVisibility(View.GONE);
            }
        });

    }

    @Override
    public void onBackPressed() {
        if (wView.canGoBack()) {
            wView.goBack();
        } else {
            if ((System.currentTimeMillis() - exitTime) > 2000) {
                Toast.makeText(getApplicationContext(), "再按一次退出
                        Toast.LENGTH_SHORT).show();
                exitTime = System.currentTimeMillis();
            } else {
                finish();
            }

        }
```



```
        }
    }
```

运行效果图：



当网页开始滚动，会呈现一个呵呵的按钮，我们点击呵呵按钮可以回到顶部！ 然后呵呵按钮会隐藏~

需求3：滚动条的问题

> 你可能用的属性如下：
>
> - setHorizontalScrollBarEnabled(false);//水平不显示
> - setVerticalScrollBarEnabled(false); //垂直不显示
> - setScrollBarStyle(View.SCROLLBARS_OUTSIDE_OVERLAY);//滚动条在WebView内侧显示
> - setScrollBarStyle(View.SCROLLBARS_INSIDE_OVERLAY)//滚动条在WebView外侧显示

需求4：设置缩放以及自适应屏幕

根据我们一般的习惯打开网页对于看不清楚的地方，我们喜欢双指来缩放网页，而WebView 则需要我们自己手动来设置这个是否支持缩放了！

只需要在加入下述代码即可：

```
WebSettings settings = wView.getSettings();
settings.setUseWideViewPort(true);//设定支持viewport
settings.setLoadWithOverviewMode(true);   //自适应屏幕
settings.setBuiltInZoomControls(true);
settings.setDisplayZoomControls(false);
settings.setSupportZoom(true);//设定支持缩放
```

使用上述代码后，进去页面就会是这样一个效果：



当我们缩放时，出现了一个恶心的问题，就是很常见的缩放控件，我们肯定是不想要的啦， 那么加上下面句代码就可以把缩放控件给隐藏掉了！

```
settings.setDisplayZoomControls(false);
```

我们也可以自行设置初始的缩放比例，只需为webView：

```
wView.setInitialScale(25);//为25%，最小缩放等级
```

嘿嘿，上面是整个网页都缩放的，不过可能有时我们仅仅是需要对字体进行缩放，那么可以 这样做：

```
settings.setTextZoom(int);
```

也可以直接通过：

```
settings.setTextSize(TextSize.LARGER);
```

来设置大小。

Android自带五个可选字体大小的值：SMALLEST(50%),SMALLER(75%),NORMAL(100%),LARGER(150%), LARGEST(200%)。

需求5.获取WebView的Cookie数据

我们都知道Cookie其实只是一个代表用户唯一标识的字符串，情景一般是：用户输入账号密码后，点击登陆，用户要拿着这个Cookie去访问服务器提供的相关服务！我们可以把cookie的获取写到onPageFinsihed的方法中，简单的可以这样写：

```
@Override
public void onPageFinished(WebView view, String url) {
    CookieManager cookieManager = CookieManager.getInstance();
    String CookieStr = cookieManager.getCookie(url);
    Log.e("HEHE", "Cookies = " + CookieStr);
    super.onPageFinished(view, url);
}
```

需求6.设置WebView的Cookie数据

嘿嘿，我们上面获取到了Cookie或者通过其他途径获得了Cookie，我们如何为WebView设置Cookie呢？我们可以在需要设置Cookie的地方加入下述代码：

```
CookieSyncManager.createInstance(MainActivity.this);
CookieManager cookieManager = CookieManager.getInstance();
cookieManager.setAcceptCookie(true);
cookieManager.setCookie(url, cookies);  //cookies是要设置的cookie字符
CookieSyncManager.getInstance().sync();
```

对了，上述代码需要写在loadUrl()之前，而且如果设置了Cookie了，尽量别再进行其他的设置 不然可能会无效，建议设置cookie的写在webView相关设置的最后面~loadUrl()之前!

4.示例代码下载：

WebViewDemo1：下载 WebViewDemo1.zip

WebViewDemo2：下载 WebViewDemo2.zip

本节小结：

> 好的，本节给大家介绍了一下WebView的基本用法，加载网页，设置缩放，字体缩放， 自适应屏幕，以及Cookie的获取以及设置；相信日常开发中还有各种奇葩的需求，不过 限于篇幅就写这么多，有idea的欢迎留言，下节我们来学习HTML5端如何通过JavaScript 来与WebView交互，并获取手机的相关数据！敬请期待~谢谢~

7.5.2 WebView和JavaScrip交互基础

本节引言：

> 在上一节中我们对Android的WebView(网页视图)进行了学习，相信已经了解了WebView的基本用法；
>
> 而本节我们要学习的就是通过：**HTML -> JS ->Java**来完成HTML5端与Android手机间的 互访！好的，话不多说，有吗有真相，让我们通过编写代码来体验这种微妙的联系吧~
>
> PS：为了方便，本节用到的HTML都是以文件的形式放到assets目录下，只需通过 loadUrl("file:///android_asset/~")即可加载对应的HTML~

1.核心步骤讲解：

首先，我们定义一个类，用于将数据暴露出来，JS通过该类暴露的方法(**Public**)来调用Android！

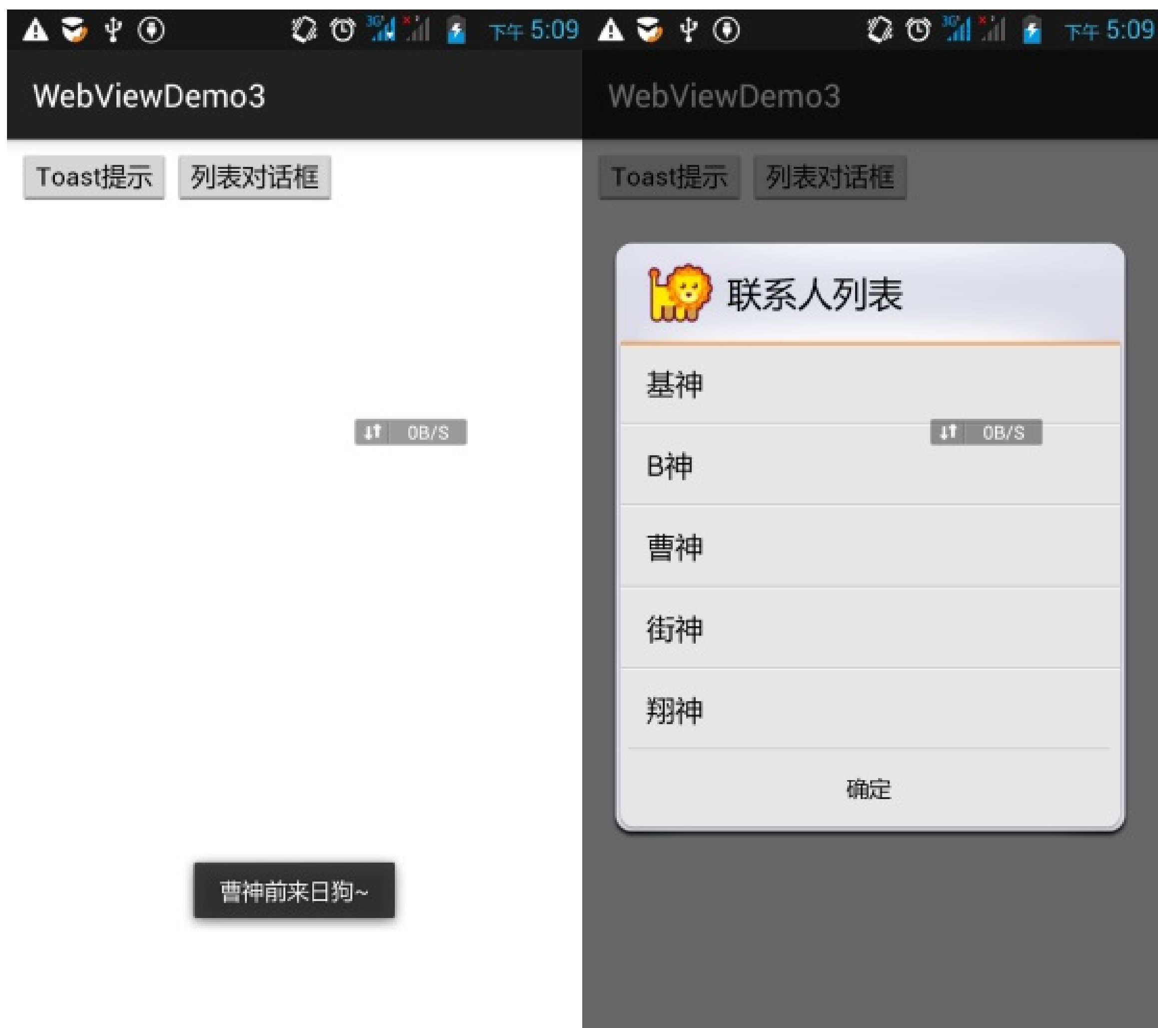
接着，我们在WebView所在页面Activity，使用下述代码:
webview.getSettings().setJavaScriptEnabled(true);
webview.addJavascriptInterface(object,"name");

然后js或者html中调用name.xxx调用对象里的暴露的方法： 比如： < input type="button" value="Toast提示" onclick="name.showToast('呵呵');"/> 另外，setJavaScriptEnabled是在**Android 4.4**以前的系统才有效！！！下一节我们会来讲解Android 4.4后 WebKit的变化以及要注意的地方！

2.使用示例讲解：

1）HTML通过JS显示Toast与普通列表的对话框

运行效果图：



代码实现：

先准备我们的HTML文件，创建好后放到assets目录下：

demo1.html:

```
<html>
<head>
    <title>Js调用Android</title>
</head>

<body>
<input type="button" value="Toast提示" onclick="myObj.showToast('曹礼
<input type="button" value="列表对话框" onclick="myObj.showDialog();
</body>
</html>
```

自定义一个Object对象，js通过该类暴露的方法来调用Android

MyObject.java:

```
/**
 * Created by Jay on 2015/9/11 0011.
 */
public class MyObject {
    private Context context;
    public MyObject(Context context) {
        this.context = context;
    }

    //将显示Toast和对话框的方法暴露给JS脚本调用
    public void showToast(String name) {
        Toast.makeText(context, name, Toast.LENGTH_SHORT).show();
    }

    public void showDialog() {
        new AlertDialog.Builder(context)
                .setTitle("联系人列表").setIcon(R.mipmap.ic_lion_ico
                .setItems(new String[]{"基神", "B神", "曹神", "街神",
                .setPositiveButton("确定", null).create().show();
    }
}
```

最后是**MainActivity.java**，启用JavaScript支持，然后通过addJavascriptInterface暴露对象~

```
public class MainActivity extends AppCompatActivity {
    private WebView wView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        wView = (WebView) findViewById(R.id.wView);
        wView.loadUrl("file:///android_asset/demo1.html");
        WebSettings webSettings = wView.getSettings();
        //①设置WebView允许调用js
        webSettings.setJavaScriptEnabled(true);
        webSettings.setDefaultTextEncodingName("UTF-8");
        //②将object对象暴露给Js,调用addjavascriptInterface
        wView.addJavascriptInterface(new MyObject(MainActivity.this
    }
}
```

2）HTML通过JS调用三种不同的对话框

运行效果图：



实现代码：

先往assets目录下塞一个html文件： **demo2.html**：

```
<html>
<head>
    <meta http-equiv = "Content-Type" content="text/html;charset=U
    <title>测试Js的三种不同对话框</title>
    <script language="JavaScript">
        function alertFun()
        {
            alert("Alert警告对话框!");
        }
        function confirmFun()
        {
            if(confirm("访问百度?"))
            {location.href = "http://www.baidu.com";}
            else alert("取消访问!");
        }
        function promptFun()
        {
            var word = prompt("Prompt对话框","请输入点什么...:");
            if(word)
            {
                alert("你输入了:"+word)
            }else{alert("呵呵,你什么都没写!");}
        }
    </script>
</head>

<body>
<p>三种对话框的使用</p>

<p>Alert对话框</p>
<p>
    <input type="submit" name = "Submit1" value="展示1" onclick="al
</p>
<p>Confirm对话框</p>
<p>
    <input type="submit" name = "Submit2" value="展示2" onclick="co
</p>
<p>Prompt对话框</p>
<p>
    <input type="submit" name = "Submit3" value="展示3" onclick="pr
</p>
</body>
</html>
```

PS：主布局和prompt布局这里就不贴了！ 直接上**MainActivity.java**:

```
public class MainActivity extends AppCompatActivity {

    private WebView wView;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        wView = (WebView) findViewById(R.id.wView);

        //获得WebSetting对象,支持js脚本,可访问文件,支持缩放,以及编码方式
        WebSettings webSettings = wView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        webSettings.setAllowFileAccess(true);
        webSettings.setBuiltInZoomControls(true);
        webSettings.setDefaultTextEncodingName("UTF-8");
        //设置WebChromeClient,处理网页中的各种js事件
        wView.setWebChromeClient(new MyWebChromeClient());
        wView.loadUrl("file:///android_asset/demo2.html");
    }

    //这里需要自定义一个类实现WebChromeClient类,并重写三种不同对话框的处理方
    //分别重写onJsAlert,onJsConfirm,onJsPrompt方法
    class MyWebChromeClient extends WebChromeClient {
        @Override
        public boolean onJsAlert(WebView view, String url, String m
                                 final JsResult result) {
            //创建一个Builder来显示网页中的对话框
            new Builder(MainActivity.this).setTitle("Alert对话框").s
                    .setPositiveButton("确定", new OnClickListener(
                        @Override
                        public void onClick(DialogInterface dialog,
                            result.confirm();
                        }
                    }).setCancelable(false).show();
            return true;
        }

        @Override
        public boolean onJsConfirm(WebView view, String url, String
                                   final JsResult result) {
            new Builder(MainActivity.this).setTitle("Confirm对话框")
                    .setPositiveButton("确定", new OnClickListener(

                        @Override
                        public void onClick(DialogInterface dialog,
                            result.confirm();
                        }
                    })
                    .setNegativeButton("取消", new DialogInterface.

                        @Override
                        public void onClick(DialogInterface dialog,
                            result.cancel();
                        }
                    }).setCancelable(false).show();
            return true;
```

```
        }

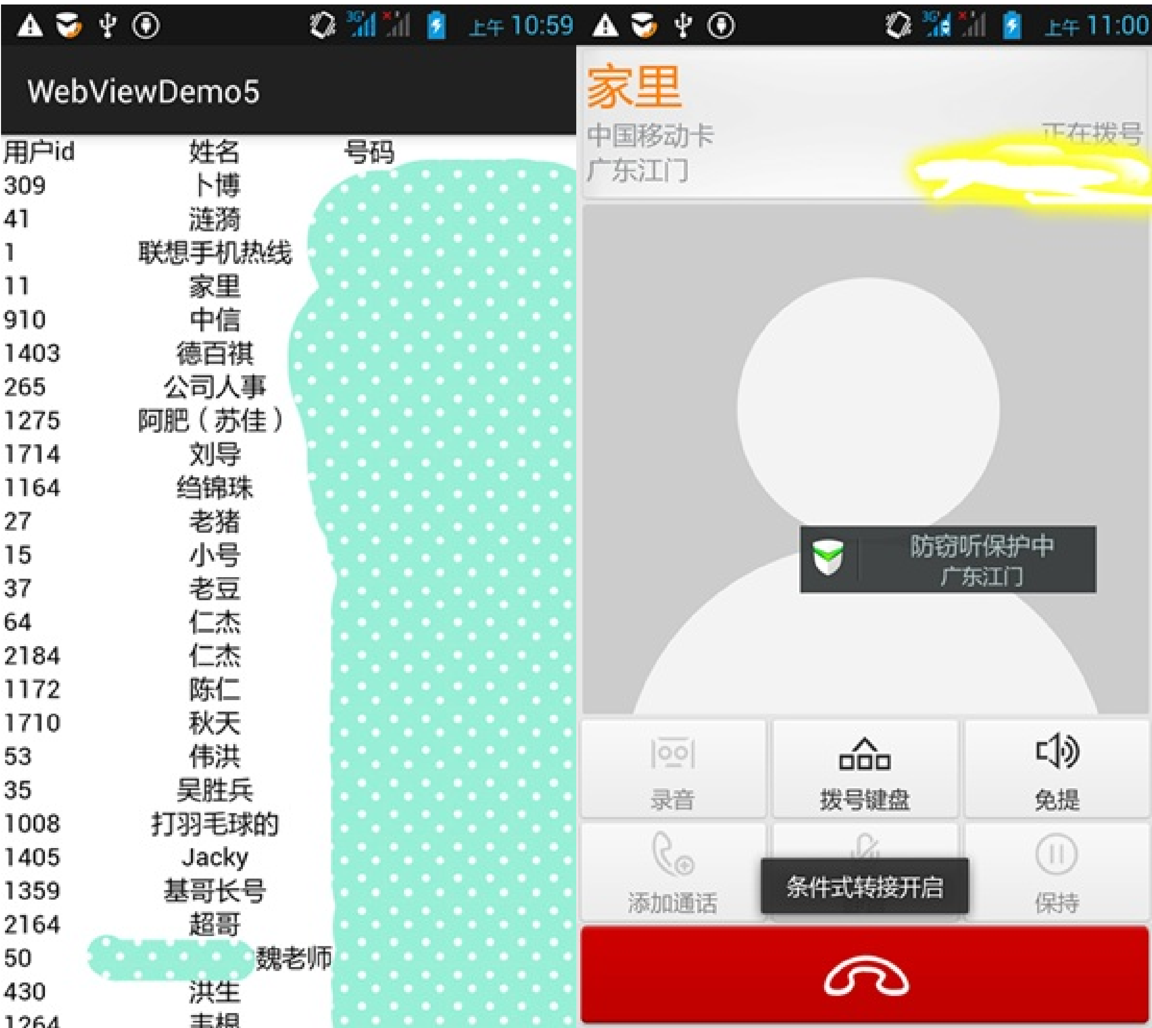
        @Override
        public boolean onJsPrompt(WebView view, String url, String
                                  String defaultValue, final JsProm
            //①获得一个LayoutInflater对象factory,加载指定布局成相应对象
            final LayoutInflater inflater = LayoutInflater.from(Mai
            final View myview = inflater.inflate(R.layout.prompt_vi
            //设置TextView对应网页中的提示信息,edit设置来自于网页的默认文字
            ((TextView) myview.findViewById(R.id.text)).setText(mes
            ((EditText) myview.findViewById(R.id.edit)).setText(def
            //定义对话框上的确定按钮
            new Builder(MainActivity.this).setTitle("Prompt对话框").
                    .setPositiveButton("确定", new OnClickListener(
                        @Override
                        public void onClick(DialogInterface dialog,
                            //单击确定后取得输入的值,传给网页处理
                            String value = ((EditText) myview.findV
                            result.confirm(value);
                        }
                    })
                    .setNegativeButton("取消", new OnClickListener(
                        @Override
                        public void onClick(DialogInterface dialog,
                            result.cancel();
                        }
                    }).show();
            return true;
        }

    }
}
```

3.HTML通过JS读取Android联系人并显示

> 该代码实现的是通过js读取Android手机中联系列表,然后显示到HTML中 当我们点击某个电话号码时,会直接跳转到拨号页面 实现关键： 利用onload()在网页加载的时候加载相应的js脚本,而js脚本中定义的一个函数是 取出传递过来的对象,获取里面的数据,通过for循环以单元行的形式打印出来！

运行效果图：



实现代码：

往assets文件夹下编写要给**demo3.html**文件，内容如下：

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ut
    <title>显示获取的联系人列表</title>
    <script language="JavaScript">
        function show(jsondata)
        {
            //将传递过来的Json转换为对象
            var jsonobjs = eval(jsondata);
            //获取下面定义的表格
            var table = document.getElementById("PersonTable");
            //遍历上面创建的Json对象,将每个对象添加为
            //表格中的一行,而它的每个属性作为一列
            for(var i = 0;i < jsonobjs.length;i++)
            {
                //添加一行,三个单元格:
                var tr = table.insertRow(table.rows.length);
                var td1 = tr.insertCell(0);
                var td2 = tr.insertCell(1);
                td2.align = "center";
                var td3 = tr.insertCell(2);
                //设置单元格的内容和属性
                //其中innerHTML为设置或者获取位于对象起始和结束标签内的HTM
                //jsonobjs[i]为对象数组中的第i个对象
                td1.innerHTML = jsonobjs[i].id;
                td2.innerHTML = jsonobjs[i].name;
                //为现实的内容添加超链接,超链接会调用Java代码中的
                //call方法并且把内容作为参数传递过去
                td3.innerHTML = "<a href = 'javascript:sharp.call(
                +jsonobjs[i].phone + "</a>";;
            }
        }
    </script>
</head>

<!-- onload指定该页面被加载时调用的方法,这里调用的是Java代码中的contactlis
<body style="margin:0px; background-color:#FFFFFF; color:#000000;"
<!--定义一个表格-->
<table border = "0" width = "100%" id = "PersonTable" cellspacing =
    <tr>
        <td width = "15%">用户id</td>
        <td align = "center">姓名</td>
        <td width = "15%">号码</td>
    </tr>
</table>
</body>
</html>
```

再写一个业务类**Contact.java**：

```
/**
 * Created by Jay on 2015/9/11 0011.
 */
public class Contact {
    private String id;
    private String name;
    private String phone;

    public Contact(){}

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    @Override
    public String toString() {
        return this.id + "~" + this.name + "~" + this.phone;
    }
}
```

再写一个业务类**Contact.java**：

```
public class MainActivity extends AppCompatActivity {

    private WebView wView;

    @SuppressLint("JavascriptInterface")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        //设置WebView的相关设置,依次是:
        //支持js,不保存表单,不保存密码,不支持缩放
        //同时绑定Java对象
        wView = (WebView) findViewById(R.id.wView);
        wView.getSettings().setJavaScriptEnabled(true);
        wView.getSettings().setSaveFormData(false);
        wView.getSettings().setSavePassword(false);
        wView.getSettings().setSupportZoom(false);
        wView.getSettings().setDefaultTextEncodingName("UTF-8");
        wView.addJavascriptInterface(new SharpJS(), "sharp");
        wView.loadUrl("file:///android_asset/demo3.html");
    }

    //自定义一个Js的业务类,传递给JS的对象就是这个,调用时直接javascript:shar
    public class SharpJS {
        public void contactlist() {
            try {
                System.out.println("contactlist()方法执行了！");
                String json = buildJson(getContacts());
                wView.loadUrl("javascript:show('" + json + "')");
            } catch (Exception e) {
                System.out.println("设置数据失败" + e);
            }
        }

        public void call(String phone) {
            System.out.println("call()方法执行了！");
            Intent it = new Intent(Intent.ACTION_CALL, Uri.parse("t
            startActivity(it);
        }
    }

    //将获取到的联系人集合写入到JsonObject对象中,再添加到JsonArray数组中
    public String buildJson(List<Contact> contacts)throws Exception
    {
        JSONArray array = new JSONArray();
        for(Contact contact:contacts)
        {
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("id", contact.getId());
            jsonObject.put("name", contact.getName());
            jsonObject.put("phone", contact.getPhone());
            array.put(jsonObject);
        }
        return array.toString();
    }

    //定义一个获取联系人的方法,返回的是List<Contact>的数据
    public List<Contact> getContacts()
    {
        List<Contact> Contacts = new ArrayList<Contact>();
        //①查询raw_contacts表获得联系人的id
        ContentResolver resolver = getContentResolver();
```

```
            Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_UR
            //查询联系人数据
            Cursor cursor = resolver.query(uri, null, null, null, null)
            while(cursor.moveToNext())
            {
                Contact contact = new Contact();
                //获取联系人姓名,手机号码
                contact.setId(cursor.getString(cursor.getColumnIndex(Co
                contact.setName(cursor.getString(cursor.getColumnIndex(
                contact.setPhone(cursor.getString(cursor.getColumnIndex
                Contacts.add(contact);
            }
            cursor.close();
            return Contacts;
        }
    }
```

好的，就是那么简单，但是，当你看到效果示意图，我祭出的是我大联想而非N5，就说明了，上述代码 在N5中执行不了，因为在4.4以后addJavascriptInterface()就不可以用了~至于为什么，我们会在下节课 中对WebView在4.4后的注意事项跟大家说下~

代码下载：

WebViewDemo4：下载 WebViewDemo4.zip

WebViewDemo5：下载 WebViewDemo5.zip

本节小结：

> 好的，本节我们对WebView和JavaScrip交互进行了简单的学习，有点意思是吧~，如果你会 HTML + CSS + JS，那么你可以尝试着自己创建一个HTML5端的移动APP试试~本节就说这么多， 谢谢~

7.5.3 Android 4.4后WebView的一些注意事项

本节引言：

> 本节参考原文：Android 4.4 中 WebView 使用注意事项.md
>
> 从Android 4.4开始，Android中的WebView不再是基于WebKit的，而是开始基于Chromium，这个改变 使得WebView的性能大幅提升，并且对HTML5，CSS，JavaScript有了更好的支持！
>
> 虽然chromium完全取代了以前的WebKit for Android，但Android WebView的API接口并没有变， 与老的版本完全兼容。这样带来的好处是基于WebView构建的APP，无需做任何修改， 就能享受chromium内核的高效与强大。
>
> 对于4.4后的WebView，我们需要注意下下面这些问题：

1.多线程

> 如果你在子线程中调用WebView的相关方法，而不在UI线程，则可能会出现无法预料的错误。 所以，当你的程序中需要用到多线程时候，也请使用runOnUiThread()方法来保证你关于 WebView的操作是在UI线程中进行的：

```
runOnUiThread(newRunnable(){  @Override publicvoid run(){  // Code
```

2.线程阻塞

> 永远不要阻塞UI线程，这是开发Android程序的一个真理。虽然是真理，我们却往往不自觉的 犯一些错误违背它，一个开发中常犯的错误就是：在UI线程中去等待JavaScript 的回调。 例如：

```
// This code is BAD and will block the UI thread webView.loadUrl("j
```

千万不要这样做，Android 4.4中，提供了新的Api来做这件事情。evaluateJavascript() 就是专门来异步执行JavaScript代码的。

3.evaluateJavascript() 方法

> 专门用于异步调用JavaScript方法，并且能够得到一个回调结果。

示例：

```
mWebView.evaluateJavascript(script,  new  ValueCallback<String>()
```

4.处理WebView中url的跳转

> 新版WebView对于自定义scheme的url跳转，新增了更为严格的限制条件。 当你实现了 shouldOverrideUrlLoading() 或 shouldInterceptRequest() 回调，WebView 也只会在跳转url是合法Url时才会跳转。 例如，如果你使用这样一个url ：

```
<a  href="showProfile">Show Profile</a>
```

shouldOverrideUrlLoading() 将不会被调用。

正确的使用方式是：

```
<a  href="example-app:showProfile">Show Profile</a>
```

对应的检测Url跳转的方式：

```
// The URL scheme should be non-hierarchical (no trailing slashes)
```

当然，也可以这样使用：

```
webView.loadDataWithBaseURL("example-app://example.co.uk/", HTML_DA
```

5.UserAgent变化

> 如果你的App对应的服务端程序，会根据客户端传来的UserAgent来做不同的事情，那么你需要注意 的是，新版本的WebView中，UserAgent有了些微妙的改变：

```
Mozilla/5.0  (Linux;  Android  4.4;  Nexus  4  Build/KRT16H)  Apple
```

使用**getDefaultUserAgent()**方法可以获取默认的UserAgent，也可以通过：

```
mWebView.getSettings().setUserAgentString(ua); mWebView.getSettings
```

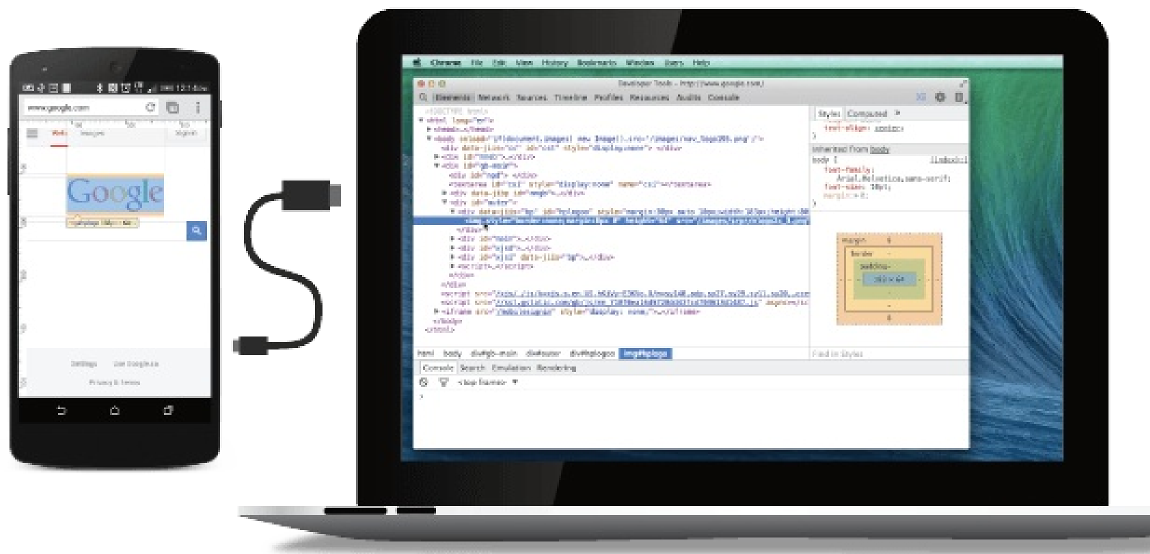
来设置和获取自定义的UserAgent。

6.使用addJavascriptInterface()的注意事项

> 从Android4.2开始。 只有添加 @JavascriptInterface 声明的Java方法才可以被JavaScript调用， 例如：

```
class  JsObject  {  @JavascriptInterface  public  String toString()
```

7.Remote Debugging

> 新版的WebView还提供了一个很厉害的功能：使用Chrome来调试你运行在WebView中的程序 具体可以看：remote-debugging PS：需要梯子~你也可以直接百度remote-debugging了解相关信息，以及如何使用！
>
> 

上一节中N5读取联系人的问题解决：

> 嘿嘿，看完上面的，我们知道在Android4.2后，只有添加@JavascriptInterface 声明的Java方法才可以被JavaScript调用，于是乎我们为之前的两个方法加上@JavascriptInterface

```
public class SharpJS {
    @JavascriptInterface
    public void contactlist() {
        try {
            System.out.println("contactlist()方法执行了！");
            String json = buildJson(getContacts());
            wView.loadUrl("javascript:show('" + json + "')");
        } catch (Exception e) {
            System.out.println("设置数据失败" + e);
        }
    }
    @JavascriptInterface
    public void call(String phone) {
        System.out.println("call()方法执行了！");
        Intent it = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + phone));
        startActivity(it);
    }
}
```

但是，加完以后，并没有和我们的预想一样，出现我们想要的联系人列表，这是为什么呢？我们通过查看Log发现下面这样一段信息：

```
W/WebView:  java.lang.Throwable: A WebView method was called on thread 'JavaBridge'. All WebView methods must be called on the same thread. (Expected Looper Loope
            at android.webkit.WebView.checkThread(WebView.java:2194)
            at android.webkit.WebView.loadUrl(WebView.java:851)
            at com.jay.webviewdemo5.MainActivity$SharpJS.contactlist(MainActivity.java:49)
            at org.chromium.base.SystemMessageHandler.nativeDoRunLoopOnce(Native Method)
            at org.chromium.base.SystemMessageHandler.handleMessage(SystemMessageHandler.java:53)
            at android.os.Handler.dispatchMessage(Handler.java:102)
            at android.os.Looper.loop(Looper.java:135)
            at android.os.HandlerThread.run(HandlerThread.java:61)
09-12 11:51:43.128    2670-2748/? I/System.out:  设置数据失败java.lang.RuntimeException: java.lang.Throwable: A WebView method was called on thread 'JavaBridge'.
09-12 11:51:43.128     759-6547/? W/ActivityManager:  getTasks: caller 10202 does not hold GET_TASKS; limiting output
```

大概的意思就是：所有的WebView方法都应该在同一个线程程中调用，而这里的contactlist方法却在 JavaBridge线程中被调用了！所以我们要要把contactlist里的东东写到同一个线程中，比如一种解决 方法，就是下面这种：

```
@JavascriptInterface
public void contactlist() {
    wView.post(new Runnable() {
        @Override
        public void run() {
            try {
                System.out.println("contactlist()方法执行了！");
                String json = buildJson(getContacts());
                wView.loadUrl("javascript:show('" + json + "')");
            } catch (Exception e) {
                System.out.println("设置数据失败" + e);
            }
        }
    });
}
```

嘿嘿，接下来运行下程序，神奇的发现，我们N5的手机联系人可以读取到了~





同理，之前第一个示例也可以这样解决~

本节小结：

> 本节跟大家走了一趟Android 4.4后WebView要注意的事项，以及一些对上一节中N5问题 的解决~相信会给大家在实际开发中使用WebView带来便利~谢谢

7.5.4 WebView文件下载

本节引言

> 本节给大家介绍的是WebView下载文件的知识点，当我们在使用普通浏览器的时候，比如UC，当我们点击到一个可供下载链接的时候，就会进行下载，WebView作为一个浏览器般的组件，当然也是支持下载，我们可以自己来写下载的流程，设置下载后的文件放哪，以什么文件名 保存，当然也可以调用其它内置的浏览器来进行下载，比如Chrome，UC等等！下面给大家演示下用法！

1.调用其它浏览器下载文件：

> 这个很简单，我们只需为WebView设置setDownloadListener，然后重写DownloadListener的 onDownloadStart，然后在里面写个Intent，然后startActivity对应的Activity即可！

关键代码如下：

```
wView.setDownloadListener(new DownloadListener(){
@Override
public void onDownloadStart(String url, String userAgent, String co
    String mimetype, long contentLength) {
        Log.e("HEHE","开始下载");
        Uri uri = Uri.parse(url);
        Intent intent = new Intent(Intent.ACTION_VIEW,uri);
        startActivity(intent);
    }
});
```

如果你手机内存在多个浏览器的话，会打开一个对话框供你选择其中一个浏览器进行下载~

2.自己写线程下载文件

> 当然，你可能不想把下载文件放到默认路径下，或者想自己定义文件名等等，你都可以自己来写 一个线程来下载文件，实现示例代码如下：

核心代码：

我们自己另外写一个下载的线程类：

DownLoadThread.java

```
/**
 * Created by Jay on 2015/9/14 0014.
 */
public class DownLoadThread implements Runnable {

    private String dlUrl;

    public DownLoadThread(String dlUrl) {
        this.dlUrl = dlUrl;
    }

    @Override
    public void run() {
        Log.e("HEHE", "开始下载~~~~~");
        InputStream in = null;
        FileOutputStream fout = null;
        try {
            URL httpUrl = new URL(dlUrl);
            HttpURLConnection conn = (HttpURLConnection) httpUrl.op
            conn.setDoInput(true);
            conn.setDoOutput(true);
            in = conn.getInputStream();
            File downloadFile, sdFile;
            if (Environment.getExternalStorageState().equals(Enviro
                Log.e("HEHE","SD卡可写");
                downloadFile = Environment.getExternalStorageDirect
                sdFile = new File(downloadFile, "csdn_client.apk");
                fout = new FileOutputStream(sdFile);
            }else{
                Log.e("HEHE","SD卡不存在或者不可读写");
            }
            byte[] buffer = new byte[1024];
            int len;
            while ((len = in.read(buffer)) != -1) {
                fout.write(buffer, 0, len);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (in != null) {
                try {
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            if (fout != null) {
                try {
                    fout.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
            }
            Log.e("HEHE", "下载完毕~~~~");
        }
    }
```

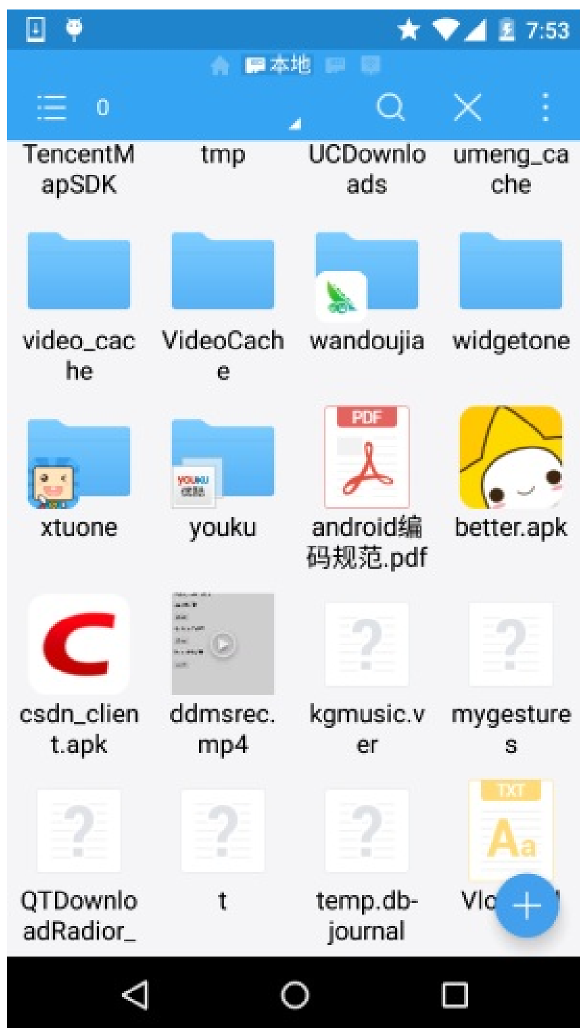
然后**MainActivity.java**中创建并启动该线程：

```
wView.setDownloadListener(new DownloadListener(){
        @Override
        public void onDownloadStart(String url, String userAgent, Strin
        String mimetype, long contentLength) {
                Log.e("HEHE","onDownloadStart被调用：下载链接：" + url);
                new Thread(new DownLoadThread(url)).start();
        }
});
```

运行结果：

```
E/HEHE :  onDownloadStart被调用: 下载链接: http://csdn-app.csdn.net/csdn.apk
E/HEHE :  开始下载~~~~~
E/HEHE :  SD卡可写
E/HEHE :  下载完毕~~~~
```

我们打开SD卡可以看到，下载好的文件已经安安静静地躺在SD卡里了：



注意事项：

好的，另外，别忘了写SD卡的读写权限以及Internet访问网络的权限：

```
<uses-permission android:name="android.permission.INTERNET"/>
<!-- 在SDCard中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FIL
<!-- 往SDCard写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_ST
```

还有，**in = conn.getInputStream();**要写在conn设置完所有东西的后面！！切记，不然什么都读不了！

本节小结：

> 本节非常简单，代码就不贴出了，其实就是setDownloadListener这个玩意，自己重写下 onDownloadStart方法来处理下载过程而已~，本节就到这里，谢谢~

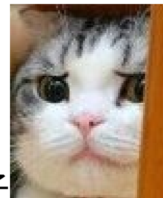
7.5.5 WebView缓存问题

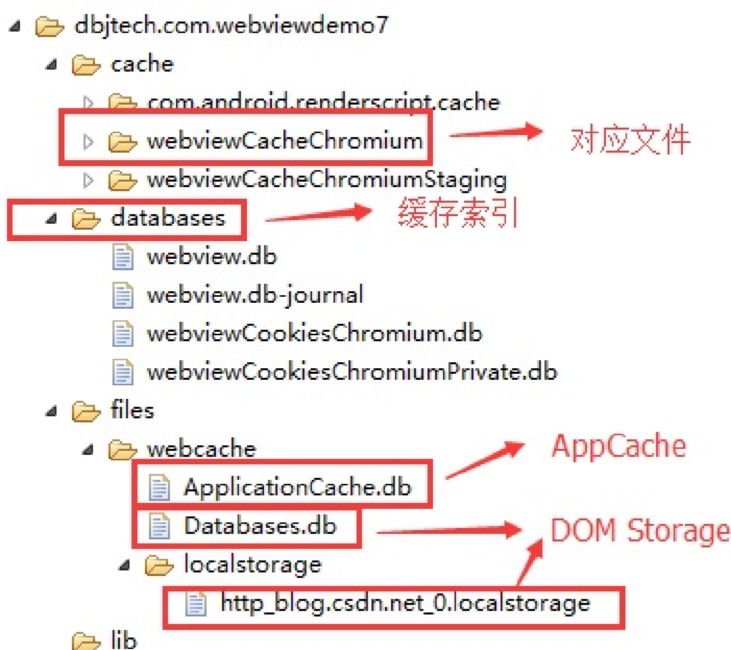
本节引言：

> 现在很多门户类信息网站，比如虎嗅，ifanr，钛媒体等等的APP，简单点说是信息阅读类的APP，很多 都是直接嵌套一个WebView用来显示相关资讯的，这可能就涉及到了WebView的缓存了！
>
> 所谓的页面缓存 就是指：保存加载一个网页时所需的HTML，JS，CSS等页面相关的数据以及其他资源，当没网的时候或者 网络状态较差的时候，加载本地保存好的相关数据！而实现这个缓存的方式有两种，一种是后台写一个 下载的Service，将文章相关的数据按自己的需求下载到数据库或者保存到相应文件夹中，然后下次加载 对应URL前先判断是否存在本地缓存，如果存在优先加载本地缓存，不存在则执行联网请求，同时缓存 相关资源，典型的如旧版本的36Kr，在进去后会先离线文章，然后再显示！
>
> 当然，本节要讲解的不是 这种自己写逻辑的方式，而是通过WebView本身自带的缓存功能来缓存页面，这种方式使用起来非常 简单，我们只需为WebView设置开启相关功能，以及设置数据库的缓存路径即可完成缓存！具体的 实现我们下面一一道来~

1.缓存的分类：

> 首先要说的一点是缓存的分类，我们缓存的数据分为：页面缓存和数据缓存
>
> - 页面缓存：加载一个网页时的html、JS、CSS等页面或者资源数据，这些缓存资源是由于浏览器 的行为而产生，开发者只能通过配置HTTP响应头影响浏览器的行为才能间接地影响到这些缓存数据。 而缓存的索引放在：/data/data/<包名>/databases 对应的文件放在：/data/data/package_name/cache/webviewCacheChromunm下
> - 数据缓存：分为AppCache和DOM Storage两种 我们开发者可以自行控制的就是这些缓存资源，
> - **AppCache**：我们能够有选择的缓冲web浏览器中所有的东西，从页面、图片到脚本、css等等。 尤其在涉及到应用于网站的多个页面上的CSS和JavaScript文件的时候非常有用。其大小目前通常是5M。 在Android上需要手动开启（setAppCacheEnabled），并设置路径（setAppCachePath）和容量 （setAppCacheMaxSize），而Android中使用**ApplicationCache.db**来保存AppCache数据！
> - **DOM Storage**：存储一些简单的用key/value对即可解决的数据，根据作用范围的不同，有Session Storage和Local Storage两种，分别用于会话级别的存储（页面关闭即消失）和本地化存储（除非主动 删除，否则数据永远不会过期）在Android中可以手动开启DOM Storage（setDomStorageEnabled）， 设置存储路径（setDatabasePath）Android中Webkit会为DOMStorage产生两个文件（my_path/localstorage/http_blog.csdn.net_0.localstorage和my_path/Databases.db）

好吧，看完上面，是不是想说一句，卧槽，什么鬼，好复杂的样子当然，不要去背，知道有这些东西就好了，实际开发用到再慢慢考究，而且我们一般只关心如何 为WebView设置缓存以及如何删除缓存！ 我们可以看下我们下面写的demo运行后的文件结构，打开DDMS的File Explorer：



嘿嘿，一目了然是吧~，对了另外还要说下几种缓存的模式：

> - **LOAD_CACHE_ONLY**: 不使用网络，只读取本地缓存数据
> - **LOAD_DEFAULT**: 根据cache-control决定是否从网络上取数据。
> - **LOAD_CACHE_NORMAL**: API level 17中已经废弃, 从API level 11开始作用同LOAD_DEFAULT模式
> - **LOAD_NO_CACHE**: 不使用缓存，只从网络获取数据.
> - **LOAD_CACHE_ELSE_NETWORK**，只要本地有，无论是否过期，或者no-cache，都使用缓存中的数据。

总结：根据以上两种模式，建议缓存策略为，判断是否有网络，有的话，使用LOAD_DEFAULT，无网络时，使用LOAD_CACHE_ELSE_NETWORK。

接下来堆码时间！

2.为WebView开启缓存功能

下面我们就来为WebView开启缓存功能，先来看下实现的效果图：

运行效果图：



流程解析： 1.进入页面后默认加载url，然后随便点击一个链接跳到第二个页面，退出APP 2.关闭wifi以及移动网络，然后重新进入，发现无网络的情况下，页面还是加载了， 打开第一个链接也可以加载，打开其他链接就发现找不到网页！ 3.点击清除缓存，把应用关闭，重新进入，发现页面已经打不开！

接下来是代码实现：**MainActivity.java**:

```
public class MainActivity extends AppCompatActivity {

    private WebView wView;
    private Button btn_clear_cache;
    private Button btn_refresh;
    private static final String APP_CACHE_DIRNAME = "/webcache"; //
    private static final String URL = "http://blog.csdn.net/coder_p

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        wView = (WebView) findViewById(R.id.wView);
        btn_clear_cache = (Button) findViewById(R.id.btn_clear_cach
        btn_refresh = (Button) findViewById(R.id.btn_refresh);
        wView.loadUrl(URL);
        wView.setWebViewClient(new WebViewClient() {
            //设置在webView点击打开的新网页在当前界面显示,而不跳转到新的浏览
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, S
                view.loadUrl(url);
                return true;
            }
        });
```

```
        WebSettings settings = wView.getSettings();
        settings.setJavaScriptEnabled(true);
        //设置缓存模式
        settings.setCacheMode(WebSettings.LOAD_CACHE_ELSE_NETWORK);
        // 开启DOM storage API 功能
        settings.setDomStorageEnabled(true);
        // 开启database storage API功能
        settings.setDatabaseEnabled(true);
        String cacheDirPath = getFilesDir().getAbsolutePath() + APP
        Log.i("cachePath", cacheDirPath);
        // 设置数据库缓存路径
        settings.setAppCachePath(cacheDirPath);
        settings.setAppCacheEnabled(true);
        Log.i("databasepath", settings.getDatabasePath());

        btn_clear_cache.setOnClickListener(new View.OnClickListener
            @Override
            public void onClick(View v) {
                wView.clearCache(true);
            }
        });

        btn_refresh.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                wView.reload();
            }
        });
    }

    //重写回退按钮的点击事件
    @Override
    public void onBackPressed() {
        if(wView.canGoBack()){
            wView.goBack();
        }else{
            super.onBackPressed();
        }
    }
}
```

代码很简单，我们做的仅仅是开启缓存的功能，以及设置缓存模式以及缓存的数据的路径而已！

3.删除WebView的缓存数据

> 上面的示例，我们通过调用WebView的clearCache(true)方法，已经实现了对缓存的删除！除了这种方法外，还有下述方法：
>
> - **setting.setCacheMode(WebSettings.LOAD_NO_CACHE)**;
> - **deleteDatabase("WebView.db")**;和 **deleteDatabase("WebViewCache.db")**;
> - **webView.clearHistory()**;
> - **webView.clearFormData()**;
> - **getCacheDir().delete()**;
> - 手动写delete方法，循环迭代删除缓存文件夹！

当然，前面也说，我们能这直接操作的只是数据部分，而页面缓存是由于浏览器 的行为而产生，我们只能通过配置HTTP响应头影响浏览器的行为才能间接地影响到这些缓存数据。所以上述的方法仅仅是删除的数据部分的缓存！

4.示例代码下载：

WebViewDemo7.zip：下载 WebViewDemo7.zip

5.本节小结：

> 好的，本节关于WebView缓存问题就到这里，这里只是写了如何为WebView开启缓存，以及删除缓存，以后遇到再慢慢考究，这里有个映像先~嗯，就说这
>
> 
>
> 么多~谢谢
>
> 对了，差点忘了贴下本节的参考链接：
>
> Android webView 缓存 Cache + HTML5离线功能 解决
>
> Android记录25-WebView实现离线缓存阅读
>
> Android 清除WebView缓存

7.5.6 WebView处理网页返回的错误码信息

本节引言：

> 嘿嘿，假如你们公司是做HTML5端的移动APP的，就是通过WebView来显示网页的，假如你访问的网页 不存在，或者其他错误，报404，401，403，30X等错误的状态码，如果直接弹出WebView默认的错误 提示页面，可能显得不那么友好，我们可以重写WebViewClient的onReceivedError()方法来实现我们 想要的效果，一般的做法有两种，一种是：我们自己在assets目录下创建一个用于显示错误信息的 HTML页面，当发生错误，即onReceivedError()被调用的时候我们调用webView的loadUrl跳到我们 的错误页面，比如:wView.loadUrl("file:///android_asset/error.html");!又或者我们另外写 一个布局或者直接一个大大的图片，平时设置为不可见，当页面错误时，让该布局或者图片可见！ 下面我们来写个简单的示例！

1.页面错误，加载自定义网页：

运行效果图：



关键代码：

```
wView.setWebViewClient(new  WebViewClient()  {  //设置在webView点击打
```

2.页面错误，显示相应的View

运行效果图：



实现代码：

```
public  class  MainActivity  extends  AppCompatActivity  implements
```

3.示例代码下载：

WebViewDemo8.zip：下载 WebViewDemo8.zip

本节小结：

> 嗯，很简单的一个小节，哈哈，超简单是吧，另外我们还可以根据不同的errorCode来设置不同的 页面~这里就自己扩展咯，关于WebView的基本学习就到这里吧，下一节开始我们将迎来网络编程的 中一个难点：Socket网络编程，当然如果学过，自然学起来简单，没学过也没关系，小猪带你撸 Socket~
>
> 敬请期待~不贴个表情不习惯，哈哈~  谢谢~

7.6.1 Socket学习网络基础准备

本节引言：

> 为了照顾没学过Java Socket的初学者，或者说捋一捋Android开发中涉及到的网络协议相关的概念， 毕竟面试的时候，面试官来了句给我说下网络协议有几层？那么IP协议在哪层？Socket是什么鬼？ 分哪几种？TCP和UDP协议又在哪层？有什么区别...嗯，这...所以学习本节概念性的理论还是很有 必要的！那么话不多说，开始本节内容~

1.OSI七层网络模型浅析

> 当然，我们不是专业搞网络工程的，只要知道有哪些层，大概是拿来干嘛的就可以了！

OSI七层网络模型(从下往上)：

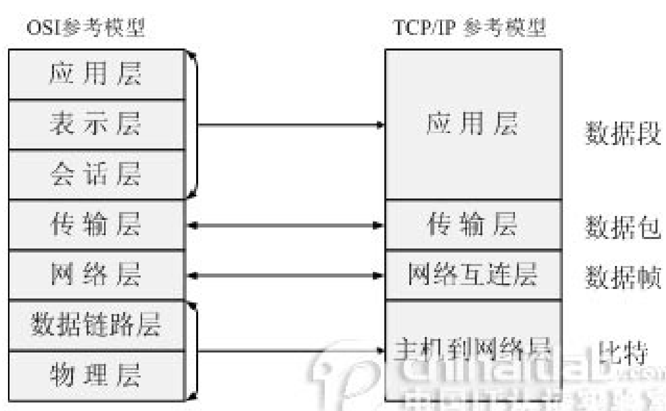
> - 物理层(**Physical**)：设备之间的数据通信提供传输媒体及互连设备，为数据传输提供可靠的 环境。可以理解为网络传输的物理媒体部分，比如网卡，网线，集线器，中继器，调制解调器等！ 在这一层，数据还没有被组织，仅作为原始的位流或电气电压处理，这一层的单位是:**bit**比特
> - 数据链路层(**Datalink**)：可以理解为数据通道，主要功能是如何在不可靠的物理线路上进行 数据的可靠传递，改层作用包括：物理地址寻址，数据的成帧，流量控制，数据检错以及重发等！ 另外这个数据链路指的是：物理层要为终端设备间的数据通信提供传输媒体及其连接。媒体是 长期的，连接是有生存期的。在连接生存期内，收发两端可以进行不等的一次或多次数据通信。 每次通信都要经过建立通信联络和拆除通信联络两过程！这种建立起来的数据收发关系~ 该层的设备有：网卡，网桥，网路交换机，另外该层的单位为：帧
> - 网络层(**Network**)：主要功能是将网络地址翻译成对应的物理地址，并决定如何将数据从发 送方路由到接收方，所谓的路由与寻径：一台终端可能需要与多台终端通信，这样就产生的了 把任意两台终端设备数据链接起来的问题！简单点说就是：建立网络连接和为上层提供服务！ 该层的设备有：路由！该层的单位为：数据包，另外IP协议就在这一层！
> - 传输层(**Transport**)：向上面的应用层提供通信服务，面向通信部分的最高层，同时也是 用户功能中的最低层。接收会话层数据，在必要时将数据进行分割，并将这些数据交给网络 层，并且保证这些数据段有效的到达对端！所以这层的单位是：数据段；而这层有两个很重要 的协议就是：**TCP**传输控制协议与**UDP**用户数据报协议，这也是本章节核心讲解的部分！
> - 会话层(**Session**)：负责在网络中的两节点之间建立、维持和终止通信。建立通信链接， 保持会话过程通信链接的畅通，同步两个节点之间的对话，决定通信是否被中断以及通信中断时 决定从何处重新发送，即不同机器上的用户之间会话的建立及管理！
> - 表示层(**Presentation**)：对来自应用层的命令和数据进行解释，对各种语法赋予相应 的含义，并按照一定的格式传送给会话层。其主要功能是"处理用户信息的表示问题，如编码、 数据格式转换和加密解密，压缩解压缩"等
> - 应用层(**Application**)：OSI参考模型的最高层，为用户的应用程序提供网络服务。 它在其他6层工作的基础上，负责完成网络中应用程序与网络操作系统之间的联系，建立与结束使用者之间的联系，并完成网络用户提出的各种网络服务及应用所需的监督、管理和服务等各种协议。此外，该层还负责协调各个应用程序间的工作。应用层为用户提供的服务和协议有：文件服务、目录服务、文件传输服务（FTP）、远程登录服务（Telnet）、电子邮件服务（E-mail）、打印服务、安全服务、网络管理服务、数据库服务等。

好的上面我们浅述了OSI七层网络模型，下面总结下：

> OSI是一个理想的模型，一般的网络系统只涉及其中的几层，在七层模型中，每一层都提供一个特殊 的网络功能，从网络功能角度观察：
>
> - 下面4层（物理层、数据链路层、网络层和传输层）主要提供数据传输和交换功能， 即以节点到节点之间的通信为主
> - 第4层作为上下两部分的桥梁，是整个网络体系结构中最关键的部分；
> - 上3层（会话层、表示层和应用层）则以提供用户与应用程序之间的信息和数据处理功能为主。
>
> 简言之，下4层主要完成通信子网的功能，上3层主要完成资源子网的功能。

——以上内容参考自：OSI七层模型详解

2.TCP/IP四层模型



> TCP/IP是一组协议的代名词，它还包括许多协议，组成了TCP/IP协议簇。TCP/IP协议簇分为四层，IP位于协议簇的第二层(对应OSI的第三层)，TCP位于协议簇的第三层 (对应OSI的第四层)。TCP/IP通讯协议采用了4层的层级结构，每一层都呼叫它的下一层所提供 的网络来完成自己的需求。这4层分别为：
>
> - 应用层：应用程序间沟通的层，如简单电子邮件传输（SMTP）、文件传输协议（FTP）、 网络远程访问协议（Telnet）等。
> - 传输层：在此层中，它提供了节点间的数据传送服务，如传输控制协议（TCP）、 用户数据报协议（UDP）等，TCP和UDP给数据包加入传输数据并把它传输到下一层中， 这一层负责传送数据，并且确定数据已被送达并接收。
> - 网络互连层：负责提供基本的数据封包传送功能，让每一块数据包都能够到达目 的主机（但不检查是否被正确接收），如网际协议（IP）。
> - 主机到网络层：对实际的网络媒体的管理，定义如何使用实际网络 （如Ethernet、Serial Line等）来传送数据。

3.TCP/UDP区别讲解

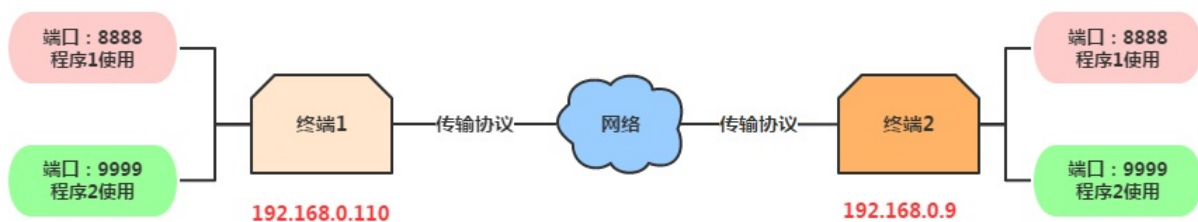
好吧，前两点侃侃而谈，只是给大家普及下OSI七层模型和TCP/IP四层模型的概念，接下来要讲的是 和我们Socket开发相关的一些概念名词了！

1）IP地址



2）端口

> 1. 用于区分不同的应用程序
>
> 2. 端口号的范围为0-65535，其中0-1023未系统的保留端口，我们的程序尽可能别使用这些端口！
>
> 3. IP地址和端口号组成了我们的Socket，Socket是网络运行程序间双向通信链路的终结点，是TCP和UDP的基础！
>
> 4. 常用协议使用的端口：HTTP:80，FTP：21，TELNET：23

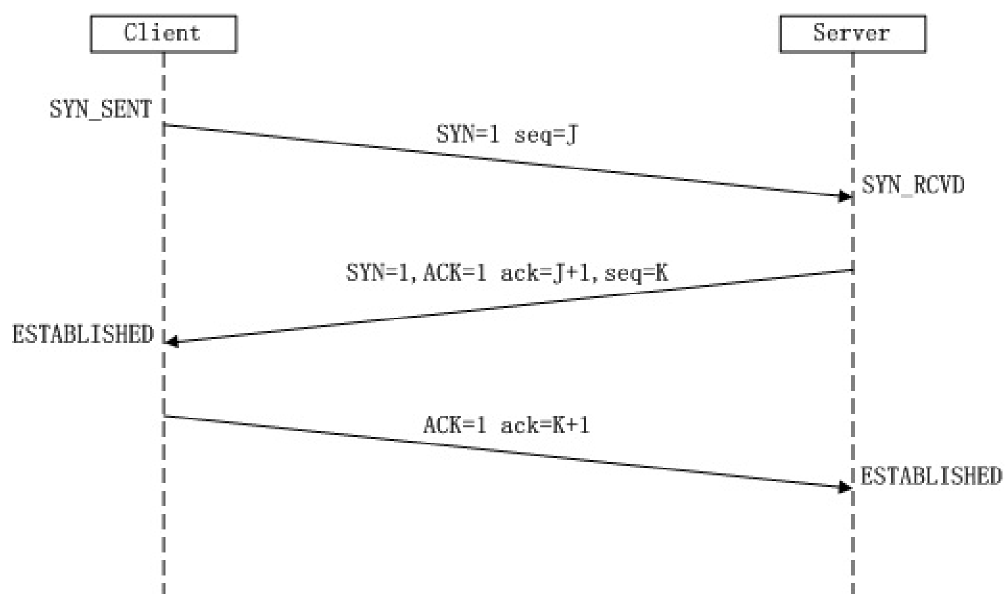


3）TCP协议与UDP协议的比较：

TCP协议流程详解:

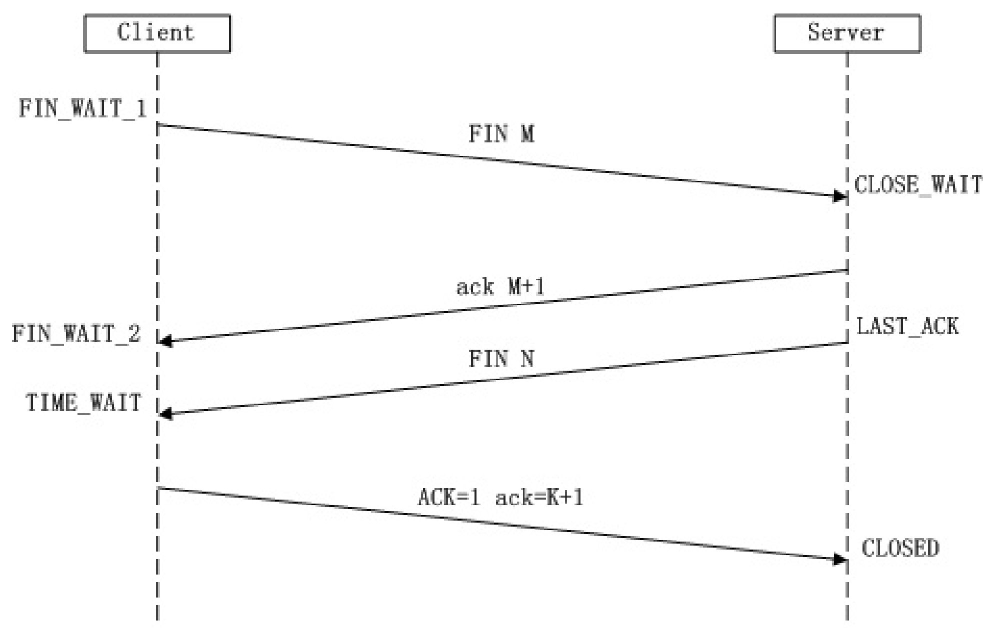
> 首先TCP/IP是一个协议簇，里面包括很多协议的。UDP只是其中的一个。之所以命名为TCP/IP协议，因为TCP,IP协议是两个很重要的协议，就用他两命名了。
>
> 下面我们来讲解TCP协议和UDP协议的区别：

TCP（Transmission Control Protocol，传输控制协议）是面向连接的协议，即在收发数据钱，都需要与对面建立可靠的链接，这也是面试经常会问到的TCP的三次握手以及TCP的四次挥手！ 三次握手： 建立一个TCP连接时，需要客户端和服务端总共发送3个包以确认连接的建立， 在Socket编程中，这一过程由客户端执行connect来触发，具体流程图如下：

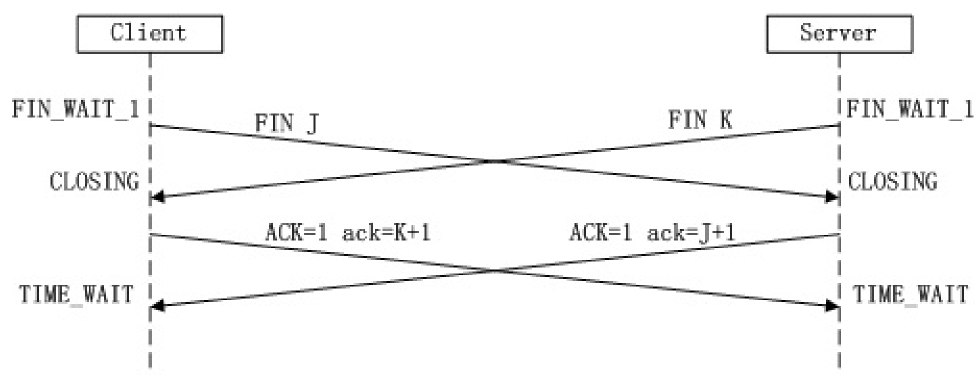


- 第一次握手：Client将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给Server， Client进入SYN_SENT状态，等待Server确认。
- 第二次握手：Server收到数据包后由标志位SYN=1知道Client请求建立连接，Server将标志位 SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给Client以确认连接请求， Server进入SYN_RCVD状态。
- 第三次握手：Client收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK 置为1，ack=K+1，并将该数据包发送给Server，Server检查ack是否为K+1，ACK是否为1，如果正确则 连接建立成功，Client和Server进入ESTABLISHED状态，完成三次握手，随后Client与Server之间可以 开始传输数据了。

四次挥手： 终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。 在Socket编程中，这一过程由客户端或服务端任一方执行close来触发，具体流程图如下：



- 第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入 FIN_WAIT_1状态
- 第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同， 一个FIN占用一个序号），Server进入CLOSE_WAIT状态。
- 第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK 状态。
- 第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。 另外也可能是同事发起主动关闭的情况：



另外还可能有一个常见的问题就是：为什么建立连接是三次握手，而关闭连接却是四次挥手呢？ 答：因为服务端在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里 发送给客户端。而关闭连接时，当收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还 能接收数据，己方也未必全部数据都发送给对方了，所以己方可以立即close，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会 分开发送。

UDP协议详解：

> **UDP**(User Datagram Protocol)用户数据报协议，非连接的协议，传输数据之前源端和终端不 建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。 在发送端，UDP传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽 的限制；在接收端，UDP把每个消息段放在队列中， 应用程序每次从队列中读一个消息段。 相比TCP就是无需建立链接，结构简单，无法保证正确性，容易丢包

——上述内容部分摘自：

TCP/IP三次握手与四次挥手

TCP和UDP的区别（转）

4.Java中对于网络提供的几个关键类：

针对不同的网络通信层次，Java给我们提供的网络功能有四大类：

- **InetAddress**： 用于标识网络上的硬件资源
- **URL**： 统一资源定位符，通过URL可以直接读取或者写入网络上的数据
- **Socket和ServerSocket**： 使用TCP协议实现网络通信的Socket相关的类
- **Datagram**： 使用UDP协议，将数据保存在数据报中，通过网络进行通信

本节我们只介绍前两个类，Socket与Datagram到TCP和UDP的章节再讲解！

~InetAddress的使用例子：

示例代码：

```
public class InetAddressTest {
    public static void main(String[] args) throws Exception{
        //获取本机InetAddress的实例：
        InetAddress address = InetAddress.getLocalHost();
        System.out.println("本机名：" + address.getHostName());
        System.out.println("IP地址：" + address.getHostAddress());
        byte[] bytes = address.getAddress();
        System.out.println("字节数组形式的IP地址：" + Arrays.toString(
        System.out.println("直接输出InetAddress对象：" + address);
    }
}
```

运行结果图：

```
本机名: coder-pig
IP地址: 172.16.
字节数组形式的IP地址: [-84, 16,
直接输出InetAddress对象: coder-pig/172.16.
```

~URL：这个就不用说了吧，忘了可以看会前面Http协议讲解那里~

本节小结：

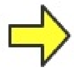
> 本节全是概念，看起来可能够呛的是把，不过看不懂也没关系，知道七层模型每层叫 什么，大概拿来干嘛，还有TCP三次握手和四次挥手，就可以了！当然，这只是为了 应付面试~实际开发我们哪会纠结这个...直接Socket是吧~嗯，下节我们就来开始学习 Android中的Socket通信~谢谢~

7.6.2 基于TCP协议的Socket通信(1)

本节引言：

> 上一节的概念课枯燥无味是吧，不过总有点收获是吧，本节开始我们来研究基于TCP协议的Socket 通信，先来了解下Socket的概念，以及Socket通信的模型，实现Socket的步骤，以及作为Socket服务 端与客户端的两位各做要做什么事情！好的，我们由浅入深来扣这个Socket吧！

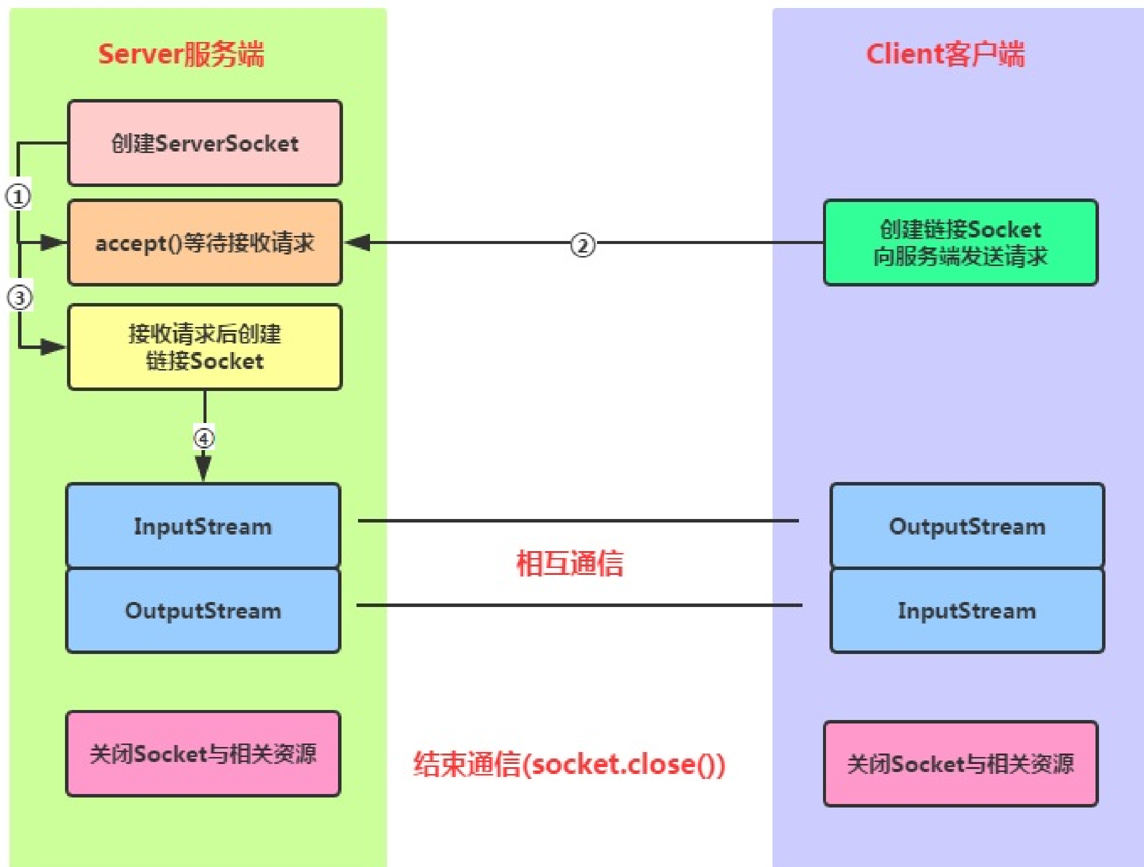
1.什么是Socket？



什么是Socket?

Socket(套接字),用来描述IP地址和端口,是通信链的句柄,应用程序可以通过Socket向网络发送请求或者应答网络请求!Socket是支持TCP/IP协议的网络通信的基本操作单元,是对网络通信过程中端点的抽象表示，包含了进行网络通信所必须的五种信息:连接所使用的协议;本地主机的IP地址;本地远程的协议端口;远地主机的IP地址以及远地进程的协议端口

2.Socket通信模型：



Socket通信实现步骤解析：

> **Step 1**：创建ServerSocket和Socket
>
> **Step 2**：打开连接到的Socket的输入/输出流
>
> **Step 3**：按照协议对Socket进行读/写操作
>
> **Step 4**：关闭输入输出流，以及Socket

好的，我们接下来写一个简单的例子，开启服务端后，客户端点击按钮然后链接服务端， 并向服务端发送一串字符串，表示通过Socket链接上服务器~

3.Socket服务端的编写：

服务端要做的事有这些：

> **Step 1**：创建ServerSocket对象，绑定监听的端口
>
> **Step 2**：调用accept()方法监听客户端的请求
>
> **Step 3**：连接建立后，通过输入流读取客户端发送的请求信息
>
> **Step 4**：通过输出流向客户端发送响应信息
>
> **Step 5**：关闭相关资源

代码实现：

直接在Eclipse下创建一个Java项目，然后把Java代码贴进去即可！

```
public class SocketServer {
    public static void main(String[] args) throws IOException {
        //1.创建一个服务器端Socket，即ServerSocket，指定绑定的端口，并监听
        ServerSocket serverSocket = new ServerSocket(12345);
        InetAddress address = InetAddress.getLocalHost();
        String ip = address.getHostAddress();
        Socket socket = null;
        //2.调用accept()等待客户端连接
        System.out.println("~~~服务端已就绪，等待客户端接入~，服务端ip地址
        socket = serverSocket.accept();
        //3.连接后获取输入流，读取客户端信息
        InputStream is=null;
        InputStreamReader isr=null;
        BufferedReader br=null;
        OutputStream os=null;
        PrintWriter pw=null;
        is = socket.getInputStream();     //获取输入流
        isr = new InputStreamReader(is,"UTF-8");
        br = new BufferedReader(isr);
        String info = null;
        while((info=br.readLine())!=null){//循环读取客户端的信息
            System.out.println("客户端发送过来的信息" + info);
        }
        socket.shutdownInput();//关闭输入流
        socket.close();
    }
}
```

然后我们把代码run起来，控制台会打印：

```
SocketServer [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年9月16日
~~~服务端已就绪，等待客户端接入~，服务端ip地址: 172.16.2.54
```

好的，接下来到Android客户端了！

4.Socket客户端的编写：

客户端要做的事有这些：

> **Step 1**：创建Socket对象，指明需要链接的服务器的地址和端号
>
> **Step 2**：链接建立后，通过输出流向服务器发送请求信息
>
> **Step 3**：通过输出流获取服务器响应的信息
>
> **Step 4**：关闭相关资源

代码实现：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn_accept = (Button) findViewById(R.id.btn_accept);
        btn_accept.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        new Thread() {
            @Override
            public void run() {
                try {
                    acceptServer();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }.start();
    }

    private void acceptServer() throws IOException {
        //1.创建客户端Socket，指定服务器地址和端口
        Socket socket = new Socket("172.16.2.54", 12345);
        //2.获取输出流，向服务器端发送信息
        OutputStream os = socket.getOutputStream();//字节输出流
        PrintWriter pw = new PrintWriter(os);//将输出流包装为打印流
        //获取客户端的IP地址
        InetAddress address = InetAddress.getLocalHost();
        String ip = address.getHostAddress();
        pw.write("客户端：~" + ip + "~ 接入服务器！！");
        pw.flush();
        socket.shutdownOutput();//关闭输出流
        socket.close();
    }
}
```

因为Android不允许在主线程(UI线程)中做网络操作，所以这里需要我们自己 另开一个线程来连接Socket！

运行结果：

点击按钮后，服务端控制台打印：

```
<terminated> SocketServer [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.e
~~~服务端已就绪，等待客户端接入~，服务端ip地址: 172.16.2.54
客户端发送过来的信息客户端：~127.0.0.1~ 接入服务器！！
```

5.增强版案例：小猪简易聊天室

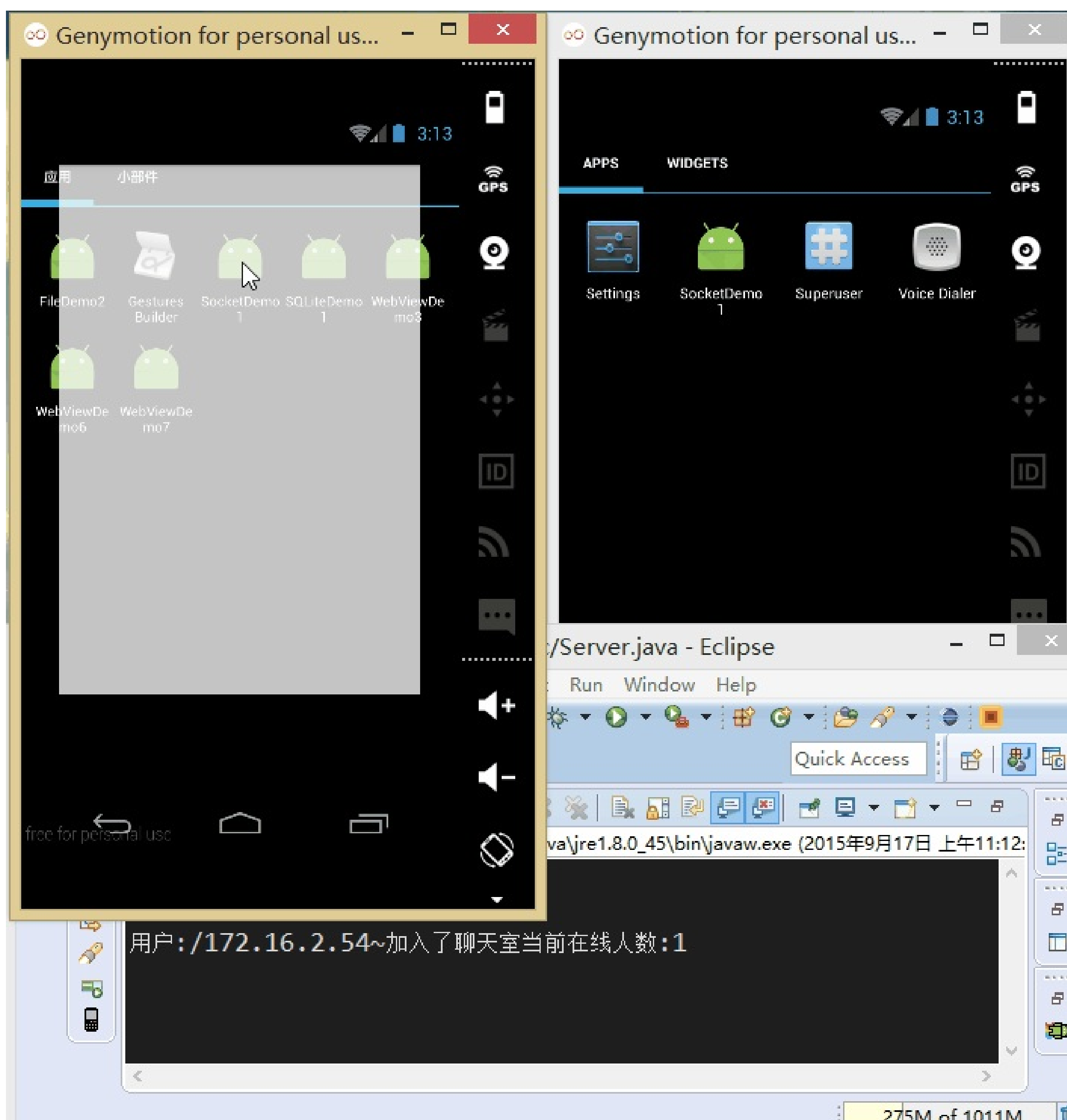
> 只是点击个按钮，然后服务器返回一串信息，肯定是很无趣的是吧，接下来我们来 搭建一个超简单的聊天室，我们需要用到线程池，存储Socket链接的集合，我们还需要 字节写一个线程，具体的我们在代码中来体会！

实现的效果图：

先把我们的服务端跑起来：

```
Server (1) [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年9月17日 上午1
服务端运行中...
```

接着把我们的程序分别跑到两台模拟器上：



接下来我们来写代码：

首先是服务端，就是将读写socket的操作放到自定义线程当中，创建ServerSocket后，循环 调用accept方法，当有新客户端接入，将socket加入集合当中，同时在线程池新建一个线程！

另外，在读取信息的方法中，对输入字符串进行判断，如果为bye字符串，将socket从集合中 移除，然后close掉！

Server.java：

```
public class Server {
    //定义相关的参数,端口,存储Socket连接的集合,ServerSocket对象
    //以及线程池
    private static final int PORT = 12345;
    private List<Socket> mList = new ArrayList<Socket>();
    private ServerSocket server = null;
    private ExecutorService myExecutorService = null;

    public static void main(String[] args) {
        new Server();
    }

    public Server()
    {
        try
        {
            server = new ServerSocket(PORT);
            //创建线程池
            myExecutorService = Executors.newCachedThreadPool();
            System.out.println("服务端运行中...\n");
            Socket client = null;
            while(true)
            {
                client = server.accept();
                mList.add(client);
                myExecutorService.execute(new Service(client));
            }

        }catch(Exception e){e.printStackTrace();}
    }

    class Service implements Runnable
    {
        private Socket socket;
        private BufferedReader in = null;
        private String msg = "";

        public Service(Socket socket) {
            this.socket = socket;
            try
            {
```

```
                in = new BufferedReader(new InputStreamReader(socke
                 msg = "用户:" +this.socket.getInetAddress() + "~加
                           +"当前在线人数:" +mList.size();
                this.sendmsg();
            }catch(IOException e){e.printStackTrace();}
        }

        @Override
        public void run() {
            try{
                while(true)
                {
                    if((msg = in.readLine()) != null)
                    {
                        if(msg.equals("bye"))
                        {
                            System.out.println("~~~~~~~~~~~~~");
                            mList.remove(socket);
                            in.close();
                            msg = "用户:" + socket.getInetAddress()
                                    + "退出:" +"当前在线人数:"+mList.
                            socket.close();
                            this.sendmsg();
                            break;
                        }else{
                            msg = socket.getInetAddress() + "   说:
                            this.sendmsg();
                        }
                    }
                }
            }catch(Exception e){e.printStackTrace();}
        }

        //为连接上服务端的每个客户端发送信息
        public void sendmsg()
        {
            System.out.println(msg);
            int num = mList.size();
            for(int index = 0;index < num;index++)
            {
                Socket mSocket = mList.get(index);
                PrintWriter pout = null;
                try {
                    pout = new PrintWriter(new BufferedWriter(
                            new OutputStreamWriter(mSocket.getOutpu
                    pout.println(msg);
                }catch (IOException e) {e.printStackTrace();}
            }
        }

    }
}
```

接着到客户端，客户端的难点在于要另外开辟线程的问题，因为Android不允许直接在 主线程中做网络操作，而且不允许在主线程外的线程操作UI，这里的做法是自己新建 一个线程，以及通过Hanlder来更新UI，实际开发不建议直接这样做！！！

布局文件:**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="小猪简易聊天室" />
    <TextView
        android:id="@+id/txtshow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/editsend"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
    <Button
        android:id="@+id/btnsend"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="发送"
        />
</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Runr

    //定义相关变量,完成初始化
    private TextView txtshow;
    private EditText editsend;
    private Button btnsend;
    private static final String HOST = "172.16.2.54";
    private static final int PORT = 12345;
    private Socket socket = null;
    private BufferedReader in = null;
    private PrintWriter out = null;
    private String content = "";
    private StringBuilder sb = null;
```

```
    //定义一个handler对象,用来刷新界面
    public Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            if (msg.what == 0x123) {
                sb.append(content);
                txtshow.setText(sb.toString());
            }
        }

        ;
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sb = new StringBuilder();
        txtshow = (TextView) findViewById(R.id.txtshow);
        editsend = (EditText) findViewById(R.id.editsend);
        btnsend = (Button) findViewById(R.id.btnsend);

        //当程序一开始运行的时候就实例化Socket对象,与服务端进行连接,获取输入
        //因为4.0以后不能再主线程中进行网络操作,所以需要另外开辟一个线程
        new Thread() {

            public void run() {
                try {
                    socket = new Socket(HOST, PORT);
                    in = new BufferedReader(new InputStreamReader(s
                    out = new PrintWriter(new BufferedWriter(new Ou
                            socket.getOutputStream())), true);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }.start();

        //为发送按钮设置点击事件
        btnsend.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                String msg = editsend.getText().toString();
                if (socket.isConnected()) {
                    if (!socket.isOutputShutdown()) {
                        out.println(msg);
                    }
                }
            }
        });
        new Thread(MainActivity.this).start();
    }
```

```
    //重写run方法,在该方法中输入流的读取
    @Override
    public void run() {
        try {
            while (true) {
                if (socket.isConnected()) {
                    if (!socket.isInputShutdown()) {
                        if ((content = in.readLine()) != null) {
                            content += "\n";
                            handler.sendEmptyMessage(0x123);
                        }
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

本节小结：

> 好的，本节给大家讲解了基于TCP的Socket通信，文中介绍了Socket通信的模型，实现了 一个简单的Socket通信例子，以及写了一个增强版的实例：小猪聊天室，相信会对刚涉及 Socket编程的你带来便利~，谢谢~

7.6.3 基于TCP协议的Socket通信(2)

本节引言：

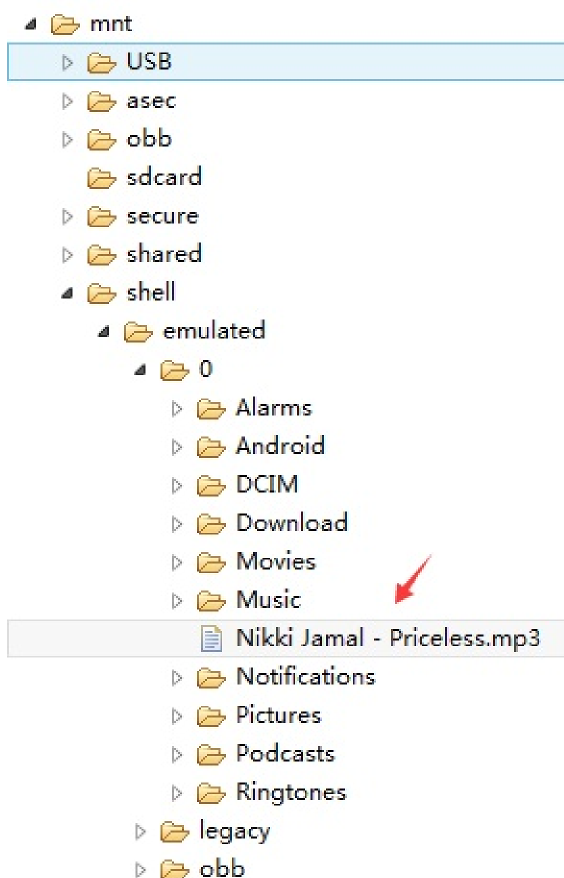
> 上节中我们给大家接触了Socket的一些基本概念以及使用方法，然后写了一个小猪简易聊天室的 Demo，相信大家对Socket有了初步的掌握，本节我们来学习下使用Socket来实现大文件的断点续传！ 这里讲解的是别人写好的一个Socket上传大文件的例子，不要求我们自己可以写出来，需要的时候会用 就好！

1.运行效果图：

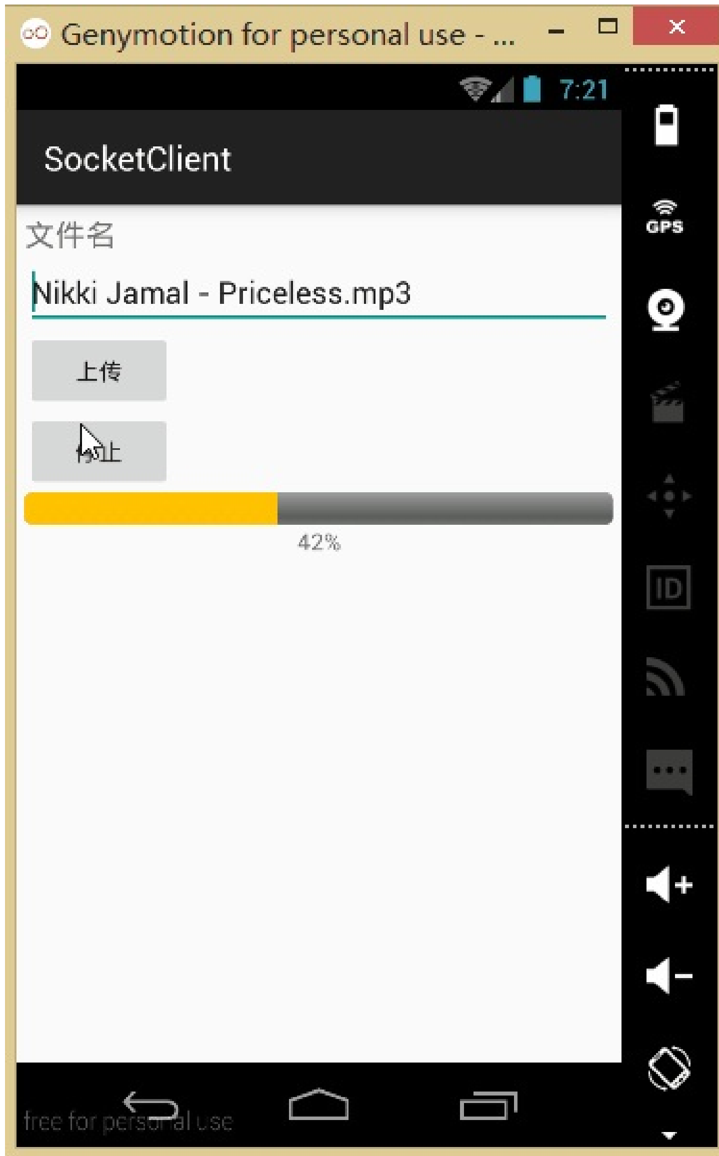
1.先把我们编写好的Socket服务端运行起来：



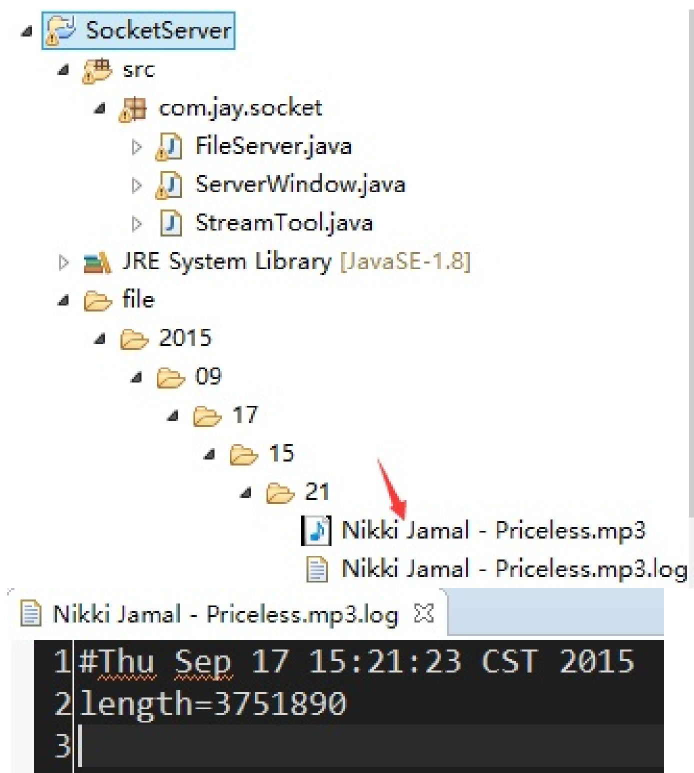
2.将一个音频文件放到SD卡根目录下：



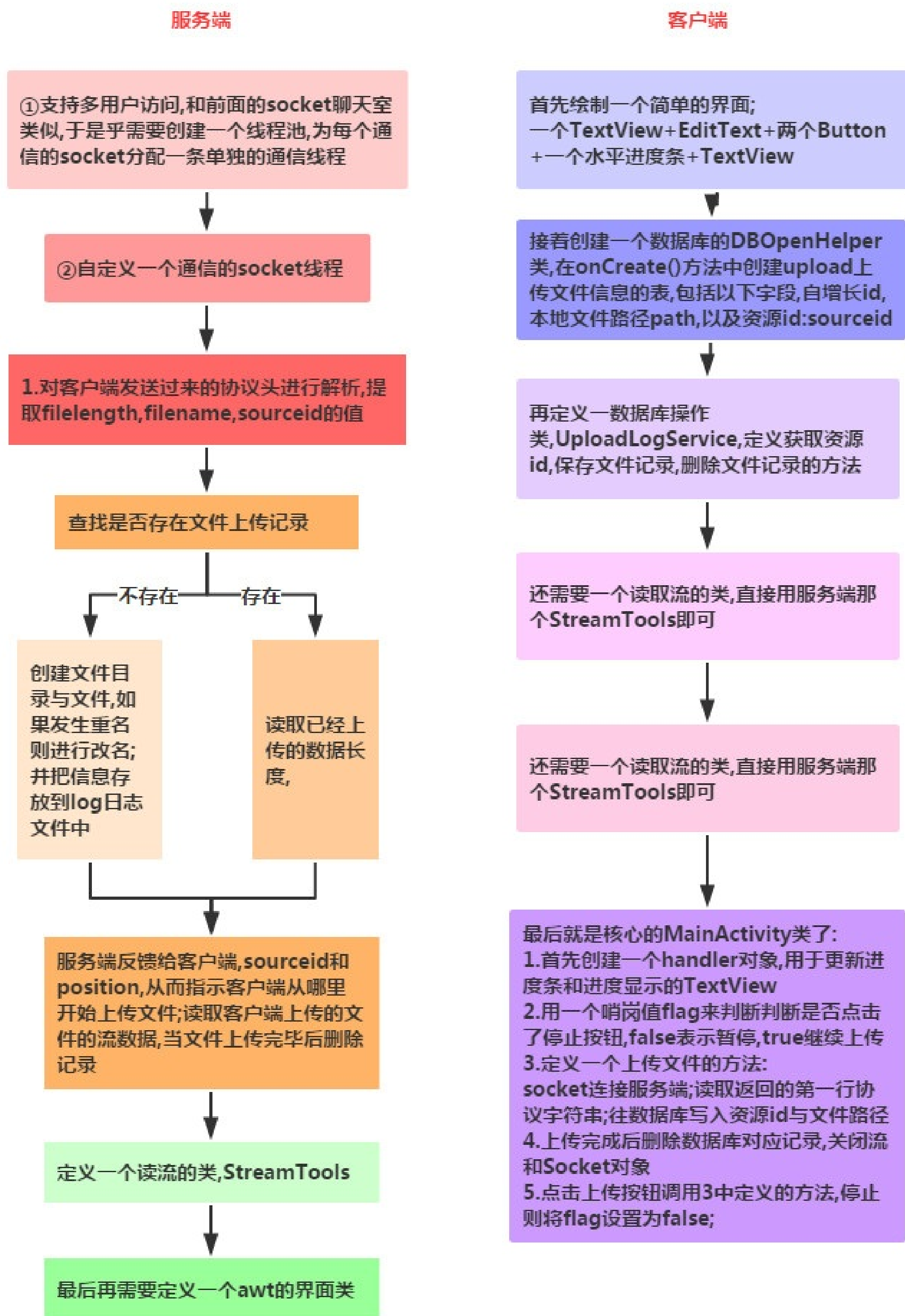
3.运行我们的客户端：



4.上传成功后可以看到我们的服务端的项目下生成一个file的文件夹，我们可以在这里找到上传的文件：.log那个是我们的日志文件


SocketServer
src
com.jay.socket
FileServer.java
ServerWindow.java
StreamTool.java
JRE System Library [JavaSE-1.8]
file
2015
09
17
15
21
Nikki Jamal - Priceless.mp3
Nikki Jamal - Priceless.mp3.log
Nikki Jamal - Priceless.mp3.log
1 #Thu Sep 17 15:21:23 CST 2015
2 length=3751890
3

2.实现流程图：


文件断点上传代码流程解析
断点上传的原理:
客户端第一次连接时想服务端发送"Content-length = xx;filename=xx.xx;sourceid="这种格式的字符串,服务端接收后会查找该文件是否有上传记录,如果有的话,返回已经上传的位置,否则返回新生产的sourceid以及position为0,客户端接收返回的字符串后再从指定位置开始上传文件,当然,协议可以由我们自己定义
服务端
客户端
①支持多用户访问,和前面的socket聊天室类似,于是乎需要创建一个线程池,为每个通信的socket分配一条单独的通信线程
②自定义一个通信的socket线程
1.对客户端发送过来的协议头进行解析,提取filelength,filename,sourceid的值
查找是否存在文件上传记录
不存在
存在
创建文件目录与文件,如果发生重名则进行改名;并把信息存放到log日志文件中
读取已经上传的数据长度,
服务端反馈给客户端,sourceid和position,从而指示客户端从哪里开始上传文件;读取客户端上传的文件的流数据,当文件上传完毕后删除记录
定义一个读流的类,StreamTools
最后再需要定义一个awt的界面类
首先绘制一个简单的界面;
一个TextView+EditText+两个Button
+一个水平进度条+TextView
接着创建一个数据库的DBOpenHelper类,在onCreate()方法中创建upload上传文件信息的表,包括以下字段,自增长id,本地文件路径path,以及资源id:sourceid
再定义一数据库操作类,UploadLogService,定义获取资源id,保存文件记录,删除文件记录的方法
还需要一个读取流的类,直接用服务端那个StreamTools即可
还需要一个读取流的类,直接用服务端那个StreamTools即可
最后就是核心的MainActivity类了:
1.首先创建一个handler对象,用于更新进度条和进度显示的TextView
2.用一个哨岗值flag来判断判断是否点击了停止按钮,false表示暂停,true继续上传
3.定义一个上传文件的方法:
socket连接服务端;读取返回的第一行协议字符串;往数据库写入资源id与文件路径
4.上传完成后删除数据库对应记录,关闭流和Socket对象
5.点击上传按钮调用3中定义的方法,停止则将flag设置为false;

3.代码示例：

先编写一个服务端和客户端都会用到的流解析类：

StreamTool.java：

```
public  class  StreamTool  {  public  static  void save(File file,
     *  读取流
     * @param inStream
     * @return 字节数组
     * @throws Exception
     */  public  static  byte[] readStream(InputStream inStream)  t
```

1）服务端的实现：

socket管理与多线程管理类：

FileServer.java：

```
public  class  FileServer  {  private  ExecutorService executorServ
     * 退出
     */  public  void quit(){  this.quit =  true;  try  { server.cl
     *  启动服务
     * @throws Exception
     */  public  void start()  throws  Exception{ server =  new  Se
              executorService.execute(new  SocketTask(socket));  }
                   outStream.write(response.getBytes());  RandomAcc
                   fileOutStream.seek(position);//指定从文件的特定位置
                       fileOutStream.write(buffer,  0, len); length
                       logFile.close();  }  if(length==fileOutStrea
        datas.put(id,  new  FileLog(id, saveFile.getAbsolutePath()
```

服务端界面类:**ServerWindow.java**：

```
public  class  ServerWindow  extends  Frame  {  private  FileServer
     * @param args
     */  public  static  void main(String[] args)  throws  IOExcept
```

2）客户端(Android端)

首先是布局文件：**activity_main.xml**：

```
<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:andro
```

因为断点续传，我们需要保存上传的进度，我们需要用到数据库，这里我们定义一个数据库 管理类：**DBOpenHelper.java**：：

```
/**
 * Created by Jay on 2015/9/17 0017.
 */ public class DBOpenHelper extends SQLiteOpenHelper { pub
```

然后是数据库操作类：**UploadHelper.java**：

```
/**
 * Created by Jay on 2015/9/17 0017.
 */ public class UploadHelper { private DBOpenHelper dbOpenHe
```

对了，别忘了客户端也要贴上那个流解析类哦，最后就是我们的**MainActivity.java**了：

```
public class MainActivity extends AppCompatActivity implement
```

最后，还有，记得往**AndroidManifest.xml**中写入这些权限哦！

```
<!-- 在SDCard中创建与删除文件权限 --> <uses-permission android:name=
```

4.代码下载：

Socket上传大文件demo

5.本节小结：

> 本节给大家介绍了基于TCP协议的Socket的另一个实例：使用Socket完成大文件的续传，相信大家对Socket的了解更进一步，嗯，下一节再写一个例子吧，两个处于同一Wifi 下的手机相互传递数据的实例吧！就说这么多，谢谢~

7.6.4 基于UDP协议的Socket通信

本节引言：

> 本节给大家带来Socket的最后一节：基于UDP协议的Socket通信，在第一节中我们已经详细地 比较了两者的区别，TCP和UDP最大的区别在于是否需要客户端与服务端建立连接后才能进行 数据传输，如果你学了前两节TCP的，传输前先开服务端，accept，等客户端接入，然后获得 客户端socket然后进行IO操作，而UDP则不用，UDP以数据报作为数据的传输载体，在进行传输时 首先要把传输的数据定义成数据报(Datagram)，在数据报中指明数据要到达的Socket(主机地址 和端口号)，然后再将数据以数据报的形式发送出去，然后就没有然后了，服务端收不收到我就 不知道了，除非服务端收到后又给我回一段确认的数据报~时间关系就不另外写Android例子了 直接上Java代码~

1.服务端实现步骤：

> **Step 1**：创建DatagramSocket，指定端口号 **Step 2**：创建DatagramPacket **Step 3**：接收客户端发送的数据信息 **Step 4**：读取数据

示例代码：

```
public class UPDServer {
    public static void main(String[] args) throws IOException {
        /*
         * 接收客户端发送的数据
         */
        // 1.创建服务器端DatagramSocket，指定端口
        DatagramSocket socket = new DatagramSocket(12345);
        // 2.创建数据报，用于接收客户端发送的数据
        byte[] data = new byte[1024];// 创建字节数组，指定接收的数据包的
        DatagramPacket packet = new DatagramPacket(data, data.lengt
        // 3.接收客户端发送的数据
        System.out.println("****服务器端已经启动，等待客户端发送数据");
        socket.receive(packet);// 此方法在接收到数据报之前会一直阻塞
        // 4.读取数据
        String info = new String(data, 0, packet.getLength());
        System.out.println("我是服务器，客户端说：" + info);

        /*
         * 向客户端响应数据
         */
        // 1.定义客户端的地址、端口号、数据
        InetAddress address = packet.getAddress();
        int port = packet.getPort();
        byte[] data2 = "欢迎您!".getBytes();
        // 2.创建数据报，包含响应的数据信息
        DatagramPacket packet2 = new DatagramPacket(data2, data2.le
        // 3.响应客户端
        socket.send(packet2);
        // 4.关闭资源
        socket.close();
    }
}
```

2.客户端实现步骤：

> **Step 1**：定义发送信息 **Step 2**：创建DatagramPacket，包含将要发送的信息 **Step 3**：创建DatagramSocket **Step 4**：发送数据

```
public class UDPClient {
    public static void main(String[] args) throws IOException {
        /*
         * 向服务器端发送数据
         */
        // 1.定义服务器的地址、端口号、数据
        InetAddress address = InetAddress.getByName("localhost");
        int port = 8800;
        byte[] data = "用户名：admin;密码：123".getBytes();
        // 2.创建数据报，包含发送的数据信息
        DatagramPacket packet = new DatagramPacket(data, data.lengt
        // 3.创建DatagramSocket对象
        DatagramSocket socket = new DatagramSocket();
        // 4.向服务器端发送数据报
        socket.send(packet);

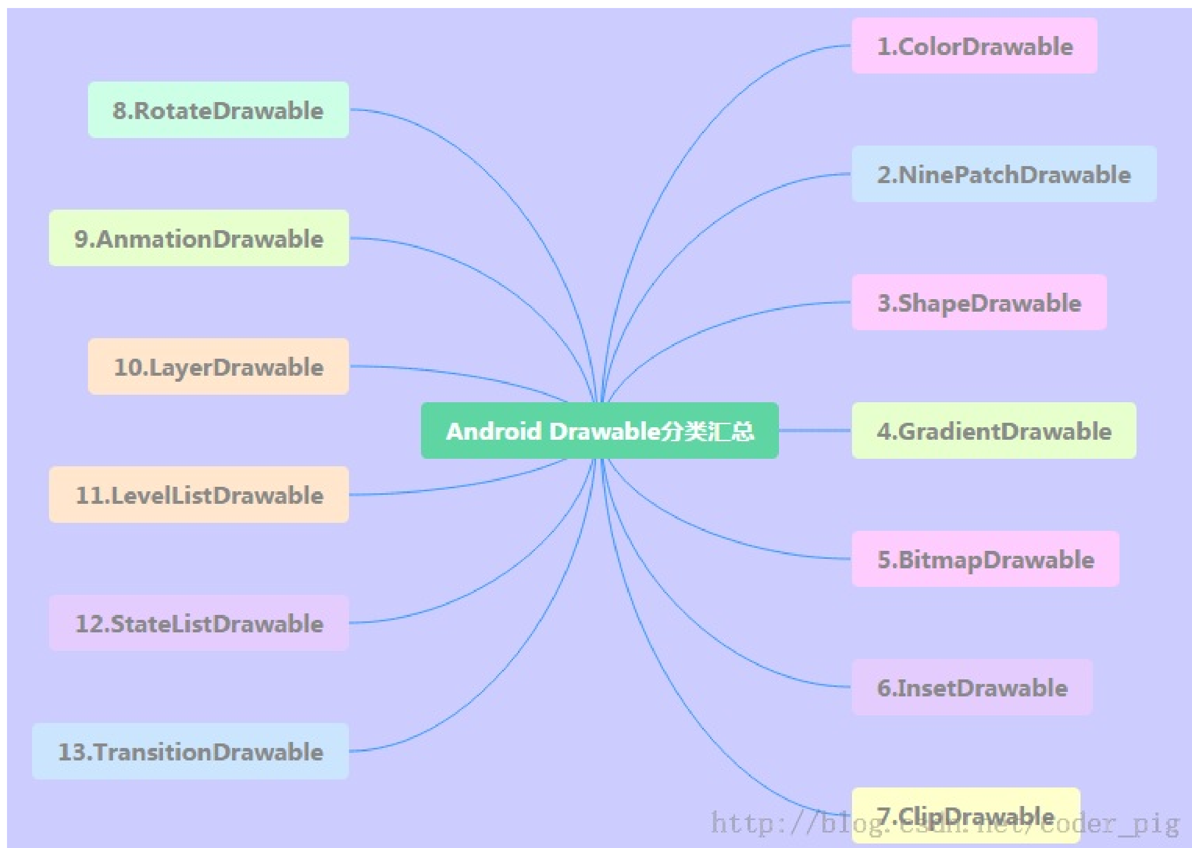
        /*
         * 接收服务器端响应的数据
         */
        // 1.创建数据报，用于接收服务器端响应的数据
        byte[] data2 = new byte[1024];
        DatagramPacket packet2 = new DatagramPacket(data2, data2.le
        // 2.接收服务器响应的数据
        socket.receive(packet2);
        // 3.读取数据
        String reply = new String(data2, 0, packet2.getLength());
        System.out.println("我是客户端，服务器说：" + reply);
        // 4.关闭资源
        socket.close();
    }
}
```

本节小结：

本节内容比较简单，无非就是将数据转换为字节，然后放到**DatagramPacket**(数据报包中)，发送的时候带上接受者的IP地址和端口号，而接收时，用一个字节数组来缓存！发送的时候需要创建一个 **DatagramSocket**(端到端通信的类)对象，然后调用send方法给接受者发送数据报包~ 本节代码来源于慕客网上的一个JavaSocket教程~有兴趣的可以看看：Java Socket应用---通信是这样练成的

8.1.1 Android中的13种Drawable小结 Part 1

本节引言：

> 从本节开始我们来学习Android中绘图与动画中的一些基础知识，为我们进阶部分的自定义 打下基础！而第一节我们来扣下Android中的Drawable！Android中给我们提供了多达13种的 Drawable，本节我们就来一个个撸一遍！
>
> 
>

Drawable资源使用注意事项

> - Drawable分为两种： 一种是我们普通的图片资源，在Android Studio中我们一般放到res/mipmap目录下， 和以前的Eclipse不一样哦！另外我们如果把工程切换成Android项目模式，我们直接 往mipmap目录下丢图片即可，AS会自动分hdpi，xhdpi...！ 另一种是我们编写的**XML**形式的**Drawable**资源，我们一般把他们放到res/drawable目录 下，比如最常见的按钮点击背景切换的Selctor！
> - 在XML我们直接通过@mipmap或者@drawable设置Drawable即可 比如: android:background = "@mipmap/iv_icon_zhu" / "@drawable/btn_back_selctor" 而在Java代码中我们可以通过Resource的getDrawable(R.mipmap.xxx)可以获得drawable资源 如果是为某个控件设置背景，比如ImageView，我们可以直接调用控件.getDrawale()同样 可以获得drawable对象！
> - Android中drawable中的资源名称有约束，必须是：**[a-z0-9_.]**（即：只能是字母数字及和.）， 而且不能以数字开头，否则编译会报错： *Invalid file name: must contain only [a-z0-9.]*！ 小写啊！！！！小写！！！小写！——重要事情说三遍~

好的，要注意的地方大概就这些，下面我们来对Android中给我们提供的13种Drawable进行学习！

1.ColorDrawable

> 最简单的一种Drawable，当我们将ColorDrawable绘制到Canvas(画布)上的时候, 会使用一种固定的颜色来填充Paint,然后在画布上绘制出一片单色区域!

1).Java中定义ColorDrawable:

```
ColorDrawable drawable = new ColorDrawable(0xffff2200);
txtShow.setBackground(drawable);
```

2).在xml中定义ColorDrawable:

```
<?xml version="1.0" encoding="utf-8"?>
<color
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#FF0000"/>
```

当然上面这些用法,其实用得不多,更多的时候我们是在res/values目录下创建一个color.xml 文件,然后把要用到的颜色值写到里面,需要的时候通过@color获得相应的值，比如：

3).建立一个color.xml文件

比如：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="material_grey_100">#fff5f5f5</color>
    <color name="material_grey_300">#ffe0e0e0</color>
    <color name="material_grey_50">#fffafafa</color>
    <color name="material_grey_600">#ff757575</color>
    <color name="material_grey_800">#ff424242</color>
    <color name="material_grey_850">#ff303030</color>
    <color name="material_grey_900">#ff212121</color>
</resources>
```

然后如果是在xml文件中话我们可以通过@color/xxx获得对应的color值 如果是在Java中:

```
int mycolor = getResources().getColor(R.color.mycolor);
btn.setBackgroundColor(mycolor);
```

ps:另外有一点要注意,如果我们在Java中直接定义颜色值的话,要加上0x,而且不能把透明度漏掉:

```
int mycolor = 0xff123456;
btn.setBackgroundColor(mycolor);
```

4).使用系统定义好的color:

比如:BLACK(黑色),BLUE(蓝色),CYAN(青色),GRAY(灰色),GREEN(绿色),RED(红色),WRITE(白色),YELLOW(黄色)! 用法：**btn.setBackgroundColor(Color.BLUE);** 也可以获得系统颜色再设置：

```
int getcolor = Resources.getSystem().getColor(android.R.color.holo_
btn.setBackgroundColor(getcolor);
```

xml中使用:**android:background="@android:color/black"**

5).利用静态方法argb来设置颜色:

> Android使用一个int类型的数据表示颜色值,通常是十六进制,即0x开头， 颜色值的定义是由透明度alpha和RGB(红绿蓝)三原色来定义的,以"#"开始,后面依次为:透明度-红-绿-蓝;eg:#RGB #ARGB #RRGGBB #AARRGGBB 每个要素都由一个字节(8 bit)来表示,所以取值范围为0~255,在xml中设置颜色可以忽略透明度,但是如果你是在Java代码中的话就需要明确指出透明度的值了,省略的话表示完全透明,这个时候 就没有效果了哦~比如:0xFF0000虽然表示红色,但是如果直接这样写,什么的没有,而应该这样写: 0xFFFF0000,记Java代码设置颜色值,需要在前面添加上透明度~ 示例:(参数依次为:透明度,红色值,绿色值,蓝色值) **txtShow.setBackgroundColor(Color.argb(0xff, 0x00, 0x00, 0x00));**

2.NiewPatchDrawable

> 就是.9图咯，在前面我们1.6 .9(九妹)图片怎么玩已经详细 的给大家讲解了一下如何制作.9图片了！Android FrameWork在显示点九图时使用了高效的 图形优化算法,我们不需要特殊的处理，就可以实现图片拉伸的自适应~ 另外在使用AS的时候要注意以下几点：
>
> - 1.点9图不能放在mipmap目录下，而需要放在drawable目录下！
> - 2.AS中的.9图，必须要有黑线，不然编译都不会通过，今早我的阿君表哥在群里说 他司的美工给了他一个没有黑线的.9图，说使用某软件制作出来的，然后在Eclipse上 是可以用的，没错是没黑线的.9，卧槽，然而我换到AS上，直接编译就不通过了！ 感觉是AS识别.9图的其中标准是需要有黑店或者黑线！另外表哥给出的一个去掉黑线的：9patch(.9)怎么去掉自己画上的黑点/黑线 具体我没试，有兴趣可以自己试试，但是黑线真的那么碍眼么...我没强迫症不觉得！ 另外还有一点就是解压别人apk，拿.9素材的时候发现并没有黑线，同样也会报错！ 想要拿出有黑线的.9素材的话，需要反编译apk而非直接解压！！！反编译前面也 介绍过了，这里就不详述了！

接着介绍两个没什么卵用的东东：

xml定义NinePatchDrawable:

```
<!--pic9.xml-->
<!--参数依次为:引用的.9图片,是否对位图进行抖动处理-->
<?xml version="1.0" encoding="utf-8"?>
<nine-patch
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/dule_pic"
    android:dither="true"/>
```

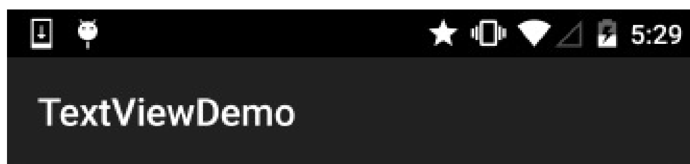
使用**Bitmap**包装**.9**图片:

```
<!--pic9.xml-->
<!--参数依次为:引用的.9图片,是否对位图进行抖动处理-->
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/dule_pic"
    android:dither="true"/>
```

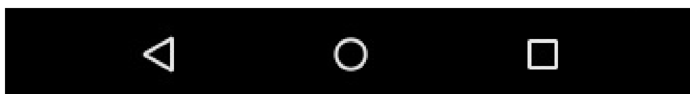
3.ShapeDrawable

> 形状的Drawable咯,定义基本的几何图形,如(矩形,圆形,线条等),根元素是 <shape../> 节点比较多， 相关的节点如下：
>
> - ① <**shape**>:
> - ~ **visible**:设置是否可见
> - ~ **shape**:形状,可选:rectangle(矩形,包括正方形),oval(椭圆,包括圆),line(线段),ring(环形)
> - ~ **innerRadiusRatio**:当shape为ring才有效,表示环内半径所占半径的比率,如果设置了innerRadius, 他会被忽略
> - ~ **innerRadius**:当shape为ring才有效,表示环的内半径的尺寸
> - ~ **thicknessRatio**:当shape为ring才有效,表环厚度占半径的比率
> - ~ **thickness**:当shape为ring才有效,表示环的厚度,即外半径与内半径的差
> - ~ **useLevel**:当shape为ring才有效,表示是否允许根据level来显示环的一部分
> - ②<**size**>:
> - ~ **width**:图形形状宽度
> - ~ **height**:图形形状高度
> - ③<**gradient**>：后面GradientDrawable再讲~
> - ④<**solid**>
> - ~ **color**:背景填充色,设置solid后会覆盖gradient设置的所有效果!!!!!!
> - ⑤<**stroke**>
> - ~ **width**:边框的宽度
> - ~ **color**:边框的颜色
> - ~ **dashWidth**:边框虚线段的长度
> - ~ **dashGap**:边框的虚线段的间距
> - ⑥<**conner**>
> - ~ **radius**:圆角半径,适用于上下左右四个角
> - ~ **topLeftRadius,topRightRadius,BottomLeftRadius,tBottomRightRadius**: 依次是左上,右上,左下,右下的圆角值,按自己需要设置!
> - ⑦<**padding**>
> - left,top,right,bottm:依次是左上右下方向上的边距!

使用示例： 2.3.1 TextView(文本框)详解





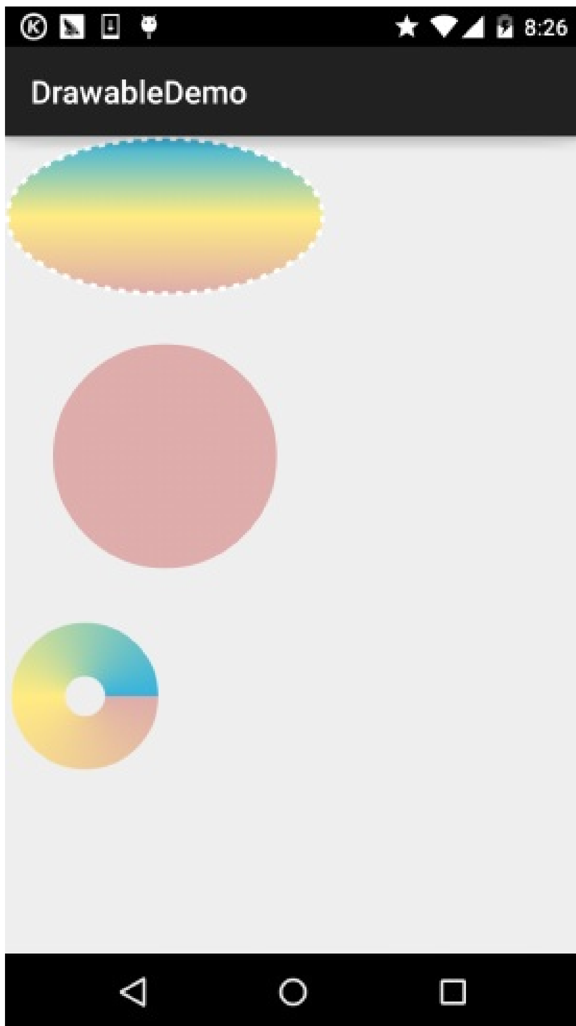


4.GradientDrawable

> 一个具有渐变区域的Drawable, 可以实现线性渐变,发散渐变和平铺渐变效果
> 核心节点：<gradient/>, 有如下可选属性：
>
> - **startColor**:渐变的起始颜色
> - **centerColor**:渐变的中间颜色
> - **endColor**:渐变的结束颜色
> - **type**:渐变类型,可选(**linear**,**radial**,**sweep**), 线性渐变(可设置渐变角度),发散渐变(中间向四周发散),平铺渐变
> - **centerX**:渐变中间亚瑟的x坐标,取值范围为:0~1
> - **centerY**:渐变中间颜色的Y坐标,取值范围为:0~1
> - **angle**:只有linear类型的渐变才有效,表示渐变角度,必须为45的倍数哦
> - **gradientRadius**:只有radial和sweep类型的渐变才有效,radial必须设置,表示渐变效果的半径
> - **useLevel**:判断是否根据level绘制渐变效果

代码示例：(三种渐变效果的演示)：

运行效果图：



先在drawable下创建三个渐变xml文件：

(线性渐变)**gradient_linear.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval" >
    <gradient
        android:angle="90"
        android:centerColor="#FFEB82"
        android:endColor="#35B2DE"
        android:startColor="#DEACAB" />

    <stroke
        android:dashGap="5dip"
        android:dashWidth="4dip"
        android:width="3dip"
        android:color="#fff" />
</shape>
```

(发散渐变)gradient_radial.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:innerRadius="0dip"
    android:shape="ring"
    android:thickness="70dip"
    android:useLevel="false" >

    <gradient
        android:centerColor="#FFEB82"
        android:endColor="#35B2DE"
        android:gradientRadius="70"
        android:startColor="#DEACAB"
        android:type="radial"
        android:useLevel="false" />

</shape>
```

(平铺渐变)gradient_sweep.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:innerRadiusRatio="8"
    android:shape="ring"
    android:thicknessRatio="3"
    android:useLevel="false" >

    <gradient
        android:centerColor="#FFEB82"
        android:endColor="#35B2DE"
        android:startColor="#DEACAB"
        android:type="sweep"
        android:useLevel="false" />

</shape>
```

调用三个drawable的**activity_main.xml**:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/txtShow1"
        android:layout_width="200dp"
        android:layout_height="100dp"
        android:background="@drawable/gradient_linear" />

    <TextView
        android:id="@+id/txtShow2"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="@drawable/gradient_radial" />

    <TextView
        android:id="@+id/txtShow3"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@drawable/gradient_sweep" />

</LinearLayout>
```

好的，就是那么简单~当然，如果想绘制更加复杂的图形的话,只用xml文件不远远不足的, 更复杂的效果则需要通过Java代码来完成,下面演示的是摘自网上的一个源码:

运行效果图：

实现代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new SampleView(this));
    }

    private static class SampleView extends View {
        private ShapeDrawable[] mDrawables;

        private static Shader makeSweep() {
            return new SweepGradient(150, 25,
                    new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF,
                    null);
        }

        private static Shader makeLinear() {
            return new LinearGradient(0, 0, 50, 50,
                    new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF
                    null, Shader.TileMode.MIRROR);
        }

        private static Shader makeTiling() {
            int[] pixels = new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF
            Bitmap bm = Bitmap.createBitmap(pixels, 2, 2,
                    Bitmap.Config.ARGB_8888);

            return new BitmapShader(bm, Shader.TileMode.REPEAT,
                    Shader.TileMode.REPEAT);
        }

        private static class MyShapeDrawable extends ShapeDrawable
            private Paint mStrokePaint = new Paint(Paint.ANTI_ALIAS

            public MyShapeDrawable(Shape s) {
                super(s);
                mStrokePaint.setStyle(Paint.Style.STROKE);
            }

            public Paint getStrokePaint() {
                return mStrokePaint;
            }

            @Override protected void onDraw(Shape s, Canvas c, Pair
                s.draw(c, p);
                s.draw(c, mStrokePaint);
            }
```

```
        }

        public SampleView(Context context) {
            super(context);
            setFocusable(true);

            float[] outerR = new float[] { 12, 12, 12, 12, 0, 0, 0,
            RectF inset = new RectF(6, 6, 6, 6);
            float[] innerR = new float[] { 12, 12, 0, 0, 12, 12, 0,

            Path path = new Path();
            path.moveTo(50, 0);
            path.lineTo(0, 50);
            path.lineTo(50, 100);
            path.lineTo(100, 50);
            path.close();

            mDrawables = new ShapeDrawable[7];
            mDrawables[0] = new ShapeDrawable(new RectShape());
            mDrawables[1] = new ShapeDrawable(new OvalShape());
            mDrawables[2] = new ShapeDrawable(new RoundRectShape(ou
                    null));
            mDrawables[3] = new ShapeDrawable(new RoundRectShape(ou
                    null));
            mDrawables[4] = new ShapeDrawable(new RoundRectShape(ou
                    innerR));
            mDrawables[5] = new ShapeDrawable(new PathShape(path, 1
            mDrawables[6] = new MyShapeDrawable(new ArcShape(45, -2

            mDrawables[0].getPaint().setColor(0xFFFF0000);
            mDrawables[1].getPaint().setColor(0xFF00FF00);
            mDrawables[2].getPaint().setColor(0xFF0000FF);
            mDrawables[3].getPaint().setShader(makeSweep());
            mDrawables[4].getPaint().setShader(makeLinear());
            mDrawables[5].getPaint().setShader(makeTiling());
            mDrawables[6].getPaint().setColor(0x88FF8844);

            PathEffect pe = new DiscretePathEffect(10, 4);
            PathEffect pe2 = new CornerPathEffect(4);
            mDrawables[3].getPaint().setPathEffect(
                    new ComposePathEffect(pe2, pe));

            MyShapeDrawable msd = (MyShapeDrawable)mDrawables[6];
            msd.getStrokePaint().setStrokeWidth(4);
        }

        @Override protected void onDraw(Canvas canvas) {

            int x = 10;
            int y = 10;
            int width = 400;
            int height = 100;
```

```
            for (Drawable dr : mDrawables) {
                dr.setBounds(x, y, x + width, y + height);
                dr.draw(canvas);
                y += height + 5;
            }
        }
    }
}
```

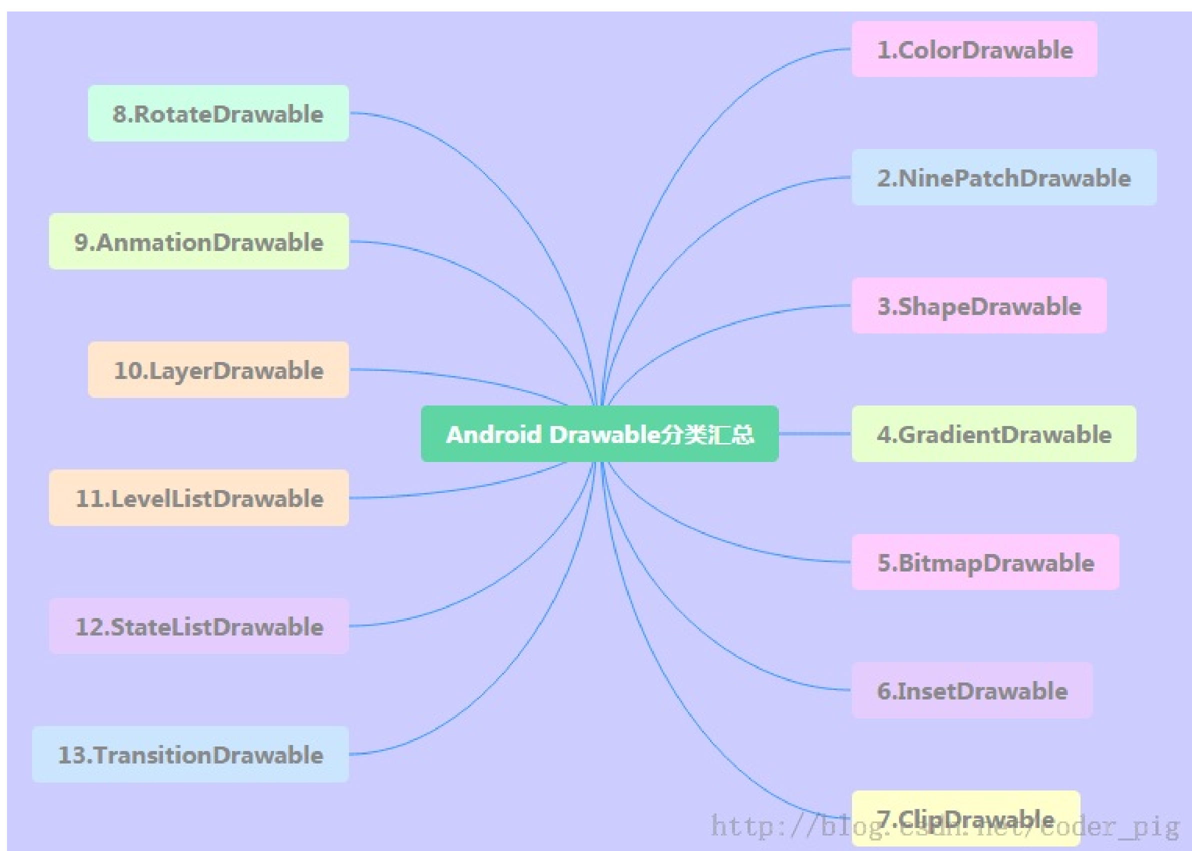
> 代码使用了ShapeDrawable和PathEffect,前者是对普通图形的包装;包括: ArcShape,OvalShape,PathShape,RectShape,RoundRectShape! 而PathEffect则是路径特效,包括:CornerPathEffect,DashPathEffect和DiscretePathEffect 可以制作复杂的图形边框... 关于这个GradoemtDrawable渐变就讲到这里,如果你对最后面这个玩意有兴趣的话,可以到: appium/android-apidemos

本节小结：

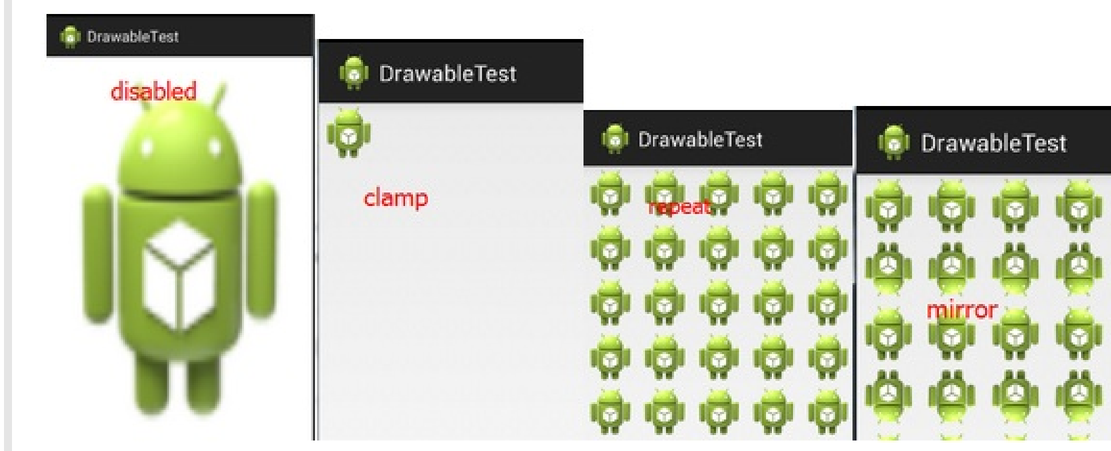
>  好的，本节就先学习ColorDrawable，NiewPatchDrawable，ShapeDrawable，GradientDrawable 四个Drawable先，当然这些都是炒冷饭，以前已经写过了，不过为了教程的完整性，还是决定 在写一遍~另外，在写完基础教程后，以前写过的一些blog会删掉！

8.1.2 Android中的13种Drawable小结 Part 2

本节引言：

> 本节我们继续来学习Android中的Drawable资源，上一节我们学习了：**ColorDrawable**；**NinePatchDrawable**；**ShapeDrawable**；**GradientDrawable**！这四个Drawable~ 而本节我们继续来学习接下来的五个Drawable，他们分别是：**BitmapDrawable**；**InsertDrawable**；**ClipDrawable**；**RotateDrawable**；**AnimationDrawable**! 还是贴下13种Drawable的导图：
>
> 
>
>
> 好的，开始本节内容~

1.BitmapDrawable

> 对Bitmap的一种封装,可以设置它包装的bitmap在BitmapDrawable区域中的绘制方式,有: 平铺填充,拉伸填或保持图片原始大小!以**<bitmap>**为根节点! 可选属性如下：
>
> - **src**:图片资源~
> - **antialias**:是否支持抗锯齿
> - **filter**:是否支持位图过滤,支持的话可以是图批判显示时比较光滑
> - **dither**:是否对位图进行抖动处理
> - **gravity**:若位图比容器小,可以设置位图在容器中的相对位置
> - **tileMode**:指定图片平铺填充容器的模式,设置这个的话,gravity属性会被忽略,有以下可选值: **disabled**(整个图案拉伸平铺),**clamp**(原图大小),**repeat**(平铺),**mirror**(镜像平铺)
>
> 对应的效果图：
>
> 
>

①XML定义BitmapDrawable:

```
<?xml version="1.0" encoding="utf-8"?>  <bitmap  xmlns:android="htt
```

②实现相同效果的Java代码:

```
BitmapDrawable bitDrawable =  new  BitmapDrawable(bitmap); bitDrawa
```

2.InsetDrawable

> 表示把一个Drawable嵌入到另外一个Drawable的内部，并且在内部留一些间距，类似与Drawable的padding属性,但**padding**表示的是**Drawable**的内容与**Drawable**本身的边距! 而**InsetDrawable**表示的是两个**Drawable**与容器之间的边距,当控件需要的背景比实际的边框 小的时候,比较适合使用InsetDrawable,比如使用这个可以解决我们自定义Dialog与屏幕之间 的一个间距问题,相信做过的朋友都知道,即使我们设置了layout_margin的话也是没用的,这个 时候就可以用到这个InsetDrawable了!只需为InsetDrawable设置一个insetXxx设置不同 方向的边距,然后为设置为Dialog的背景即可！

相关属性如下：

> - 1.**drawable**:引用的Drawable,如果为空,必须有一个Drawable类型的子节点!
> - 2.**visible**:设置Drawable是否额空间
> - 3.**insetLeft**,**insetRight**,**insetTop**,**insetBottm**:设置左右上下的边距

①XML中使用:

```
<?xml version="1.0" encoding="utf-8"?>  <inset  xmlns:android="http
```

在**Java**代码中使用：

```
InsetDrawable insetDrawable =  new  InsetDrawable(getResources()
```

使用效果图：



3.ClipDrawable

> Clip可以译为剪的意思,我们可以把ClipDrawable理解为从位图上剪下一个部分; Android中的进度条就是使用ClipDrawable来实现的,他根据设置level的值来决定剪切 区域的大小,根节点是**<clip>**

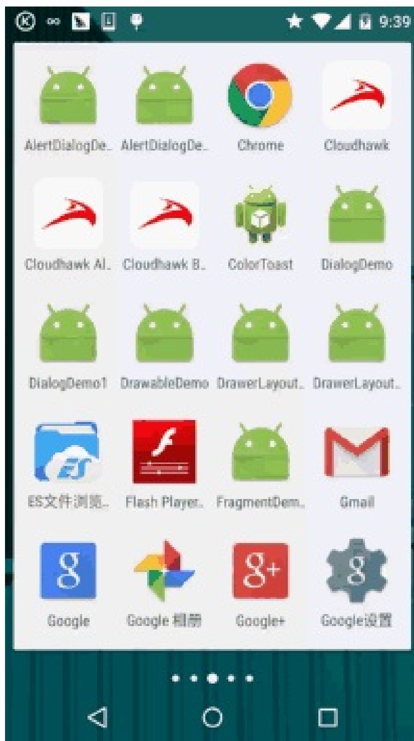
相关属性如下：

> - **clipOrietntion**:设置剪切的方向,可以设置水平和竖直2个方向
> - **gravity**:从那个位置开始裁剪
> - **drawable**:引用的drawable资源,为空的话需要有一个Drawable类型的子节点 ps:这个Drawable类型的子节点:就是在<clip里>加上这样的语句: <bitmap android:src="@drawable/test4" gravity="center">这样... </bitmap>

使用示例：

核心：通过代码修改ClipDrawable的level的值！Level的值是0~10000！

运行效果图：



代码实现：

①定义一个ClipDrawable的资源xml:

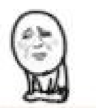
```
<?xml version="1.0" encoding="utf-8"?> <clip xmlns:android="http
```

②在activity_main主布局文件中设置一个ImageView,将src设置为clipDrawable! 记住是src哦,如果你写成了blackground的话可是会报空指针的哦!!!!

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/ar
```

③MainActivity.java通过setLevel设置截取区域大小:

```
public class MainActivity extends AppCompatActivity { private
```

好的，有点意思，妹子图别问我拿，百度上一堆哈~

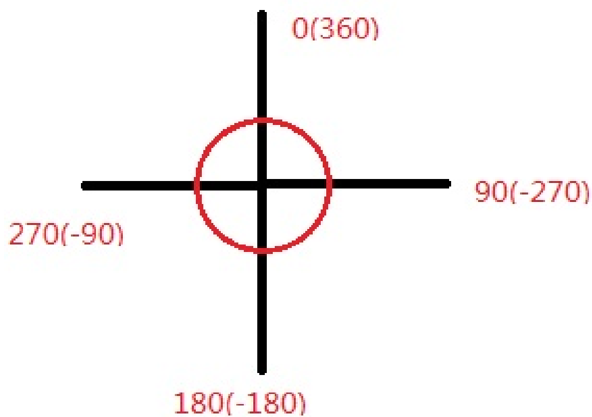
4.RotateDrawable

> 用来对Drawable进行旋转,也是通过setLevel来控制旋转的,最大值也是:10000

相关属性如下：

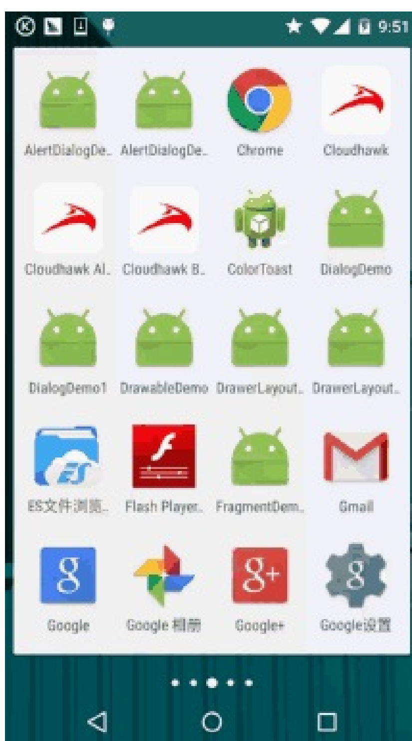
> - **fromDegrees**:起始的角度,,对应最低的level值,默认为0
> - **toDegrees**:结束角度,对应最高的level值,默认360
> - **pivotX**:设置参照点的x坐标,取值为0~1,默认是50%,即0.5
> - **pivotY**:设置参照点的Y坐标,取值为0~1,默认是50%,即0.5 ps:如果出现旋转图片显示不完全的话可以修改上述两个值解决!
> - **drawable**:设置位图资源
> - **visible**:设置drawable是否可见!

角度图如下：



使用示例：

运行效果图：



代码实现：

在第三点的clipDrawable上做一点点修改即可!

①定义一个rotateDrawable资源文件:

```
<?xml version="1.0" encoding="utf-8"?> <rotate xmlns:android="htt
```

②activity_main.xml中修改下src指向上述drawable即可,MainActivity只需要把 ClipDrawable 改成rotateDrawable即可!

```
public class MainActivity extends AppCompatActivity { private
```

5.AnimationDrawable

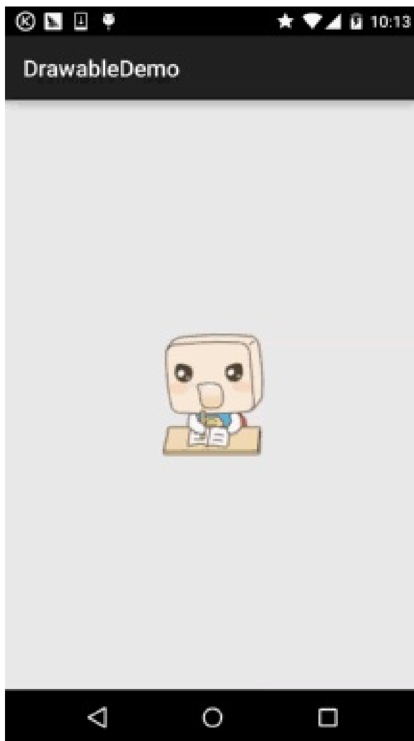
> 本节最后一个Drawable，AnimationDrawable是用来实现Android中帧动画的,就是把一系列的 Drawable，按照一定得顺序一帧帧地播放；Android中动画比较丰富,有传统补间动画,平移, 缩放等等效果,但是这里我们仅仅介绍这个AnimationDrawable实现帧动画,关于alpha,scale, translate,rotate等,后续在动画章节再进行详细的介绍~
>
> 我们这里使用**<animation-list>**作为根节点

相关属性方法:

> **oneshot**:设置是否循环播放,false为循环播放!!! **duration**:帧间隔时间,通常我们会设置为300毫秒 我们获得AniamtionDrawable实例后，需要调用它的start()方法播放动画， 另外要注意 在OnCreate()方法中调用的话,是没有任何效果的,因为View还没完成初始化,我们可以 用简单的handler来延迟播放动画!当然还有其他的方法,可见下述链接: Android AnimationDrawable运行的几种方式 使用AnimationDrawable来实现帧动画真的是非常方便的~

使用示例：

运行效果图：



代码实现：

①先定义一个**AnimationDrawable**的**xml**资源文件:

```
<?xml version="1.0" encoding="utf-8"?> <animation-list xmlns:andr
```

②**activity_main.xml**设置下**src,**然后**MainActivity**中:

```
public class MainActivity extends AppCompatActivity { private
```

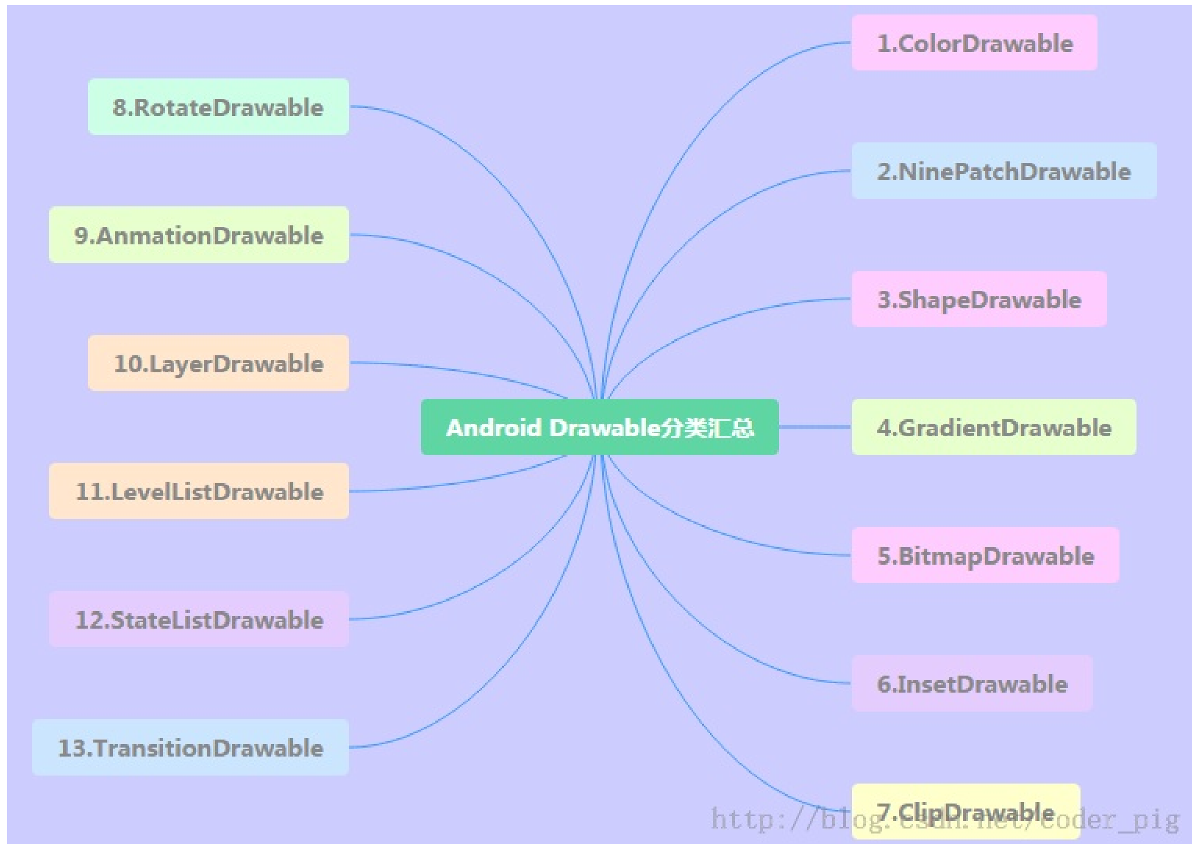
嘿嘿，超简单有木有，以后在一些需要用到帧动画的地方，直接上**AnimationDrawable**，当然，只适合于不需要进行控制的帧动画，比如上面这个就是超表下拉刷新时候的进度条素材 做成的一个简单帧动画！根据自己的需求自行拓展~

本节小结：

>  本节又介绍了另外的五个Drawable，很有趣是吧，还不快快将他们应用到你的实际开发当中~ 嘻嘻，就说这么多，谢谢!另外刚刚有读者私信我说以前的文章别删行不行，嗯，这里说下 只删除一些重复的，比如和这个雷同的那几节~当然我也会备份！删除的文章都会备份的~ 所以放心！

8.1.3 Android中的13种Drawable小结 Part 3

本节引言：

> 本节我们来把剩下的四种Drawable也学完，他们分别是：**LayerDrawable,TransitionDrawable,LevelListDrawable**和**StateListDrawable**, 依旧贴下13种Drawable的导图：
>
> 
>

1.LayerDrawable

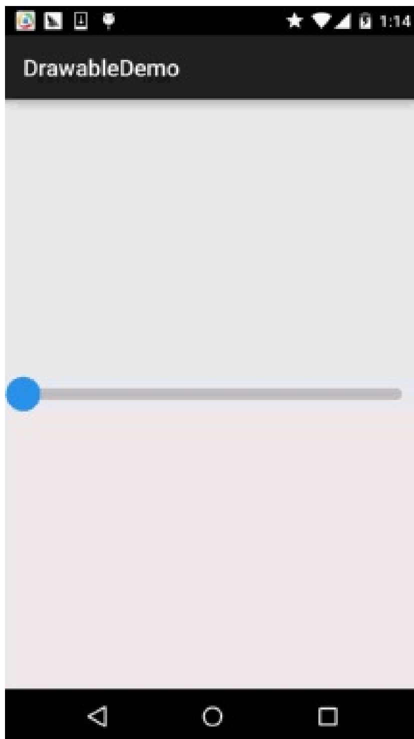
> 层图形对象，包含一个Drawable数组，然后按照数组对应的顺序来绘制他们，索引 值最大的Drawable会被绘制在最上层！虽然这些Drawable会有交叉或者重叠的区域，但 他们位于不同的层，所以并不会相互影响，以**<layer-list>**作为根节点！

相关属性如下：

> - **drawable**:引用的位图资源,如果为空徐璥有一个Drawable类型的子节点
> - **left**:层相对于容器的左边距
> - **right**:层相对于容器的右边距
> - **top**:层相对于容器的上边距
> - **bottom**:层相对于容器的下边距
> - **id**:层的id

使用示例：

运行效果图：



代码实现：

非常简单，结合前面学习的shapeDrawable和ClipDrawable：

layerList_one.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/andro
    <item android:id="@android:id/background">
        <shape android:shape="rectangle">
            <solid android:color="#C2C2C1" />
            <corners android:radius="50dp" />
        </shape>
    </item>
    <item android:id="@android:id/progress">
        <clip>
            <shape android:shape="rectangle">
                <solid android:color="#BCDA73" />
                <corners android:radius="50dp" />
            </shape>
        </clip>
    </item>
</layer-list>
```

然后在布局文件中添加一个Seekbar，内容如下：

```
<SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:indeterminateDrawable="@android:drawable/progress_i
        android:indeterminateOnly="false"
        android:maxHeight="10dp"
        android:minHeight="5dp"
        android:progressDrawable="@drawable/layerlist_one"
        android:thumb="@drawable/shape_slider" />
```

卧槽，没了？对的，就是这么点东西~说了是层图形对象，我们还可以弄个层叠图片的效果：

运行效果图：



实现代码：

层叠图片的**layerlist_two.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/andro
    <item>
        <bitmap
            android:gravity="center"
            android:src="@mipmap/ic_bg_ciwei" />
    </item>
    <item
        android:left="25dp"
        android:top="25dp">
        <bitmap
            android:gravity="center"
            android:src="@mipmap/ic_bg_ciwei" />
    </item>
    <item
        android:left="50dp"
        android:top="50dp">
        <bitmap
            android:gravity="center"
            android:src="@mipmap/ic_bg_ciwei" />
    </item>
</layer-list>
```

然后在activity_main.xml里加个ImageView，内容如下：

```
<ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/layerlist_two"/>
```

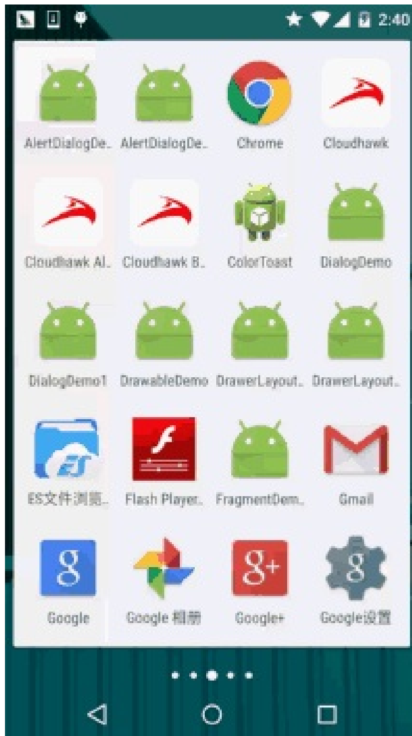
 简单好用，还等什么，快快应用到你的项目中吧~

2.TransitionDrawable

> LayerDrawable的一个子类，TransitionDrawable只管理两层的Drawable！两层！两层！并且提供了透明度变化的动画，可以控制一层Drawable过度到另一层Drawable的动画效果。根节点为**<transition>**，记住只有两个Item，多了也没用，属性和LayerDrawable差不多，我们需要调用**startTransition**方法才能启动两层间的切换动画；也可以调用**reverseTransition()**方法反过来播放：

使用示例：

运行效果图：



实现代码：

在res/drawable创建一个TransitionDrawable的xml文件

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/andro
    <item android:drawable="@mipmap/ic_bg_meizi1"/>
    <item android:drawable="@mipmap/ic_bg_meizi2"/>
</transition>
```

然后布局文件里加个ImageView，然后把src设置成上面的这个drawable 然后MainActivity.java内容如下：

```
public class MainActivity extends AppCompatActivity {
    private ImageView img_show;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img_show = (ImageView) findViewById(R.id.img_show);
        TransitionDrawable td = (TransitionDrawable) img_show.getDr
        td.startTransition(3000);
        //你可以可以反过来播放，使用reverseTransition即可~
        //td.reverseTransition(3000);
    }
}
```

另外，如果你想实现：多张图片循环的淡入淡出的效果 可参考：Android Drawable Resource学习（七）、TransitionDrawable中的示例 很简单，核心原理就是：handler定时修改Transition中两个图片！

3.LevelListDrawable

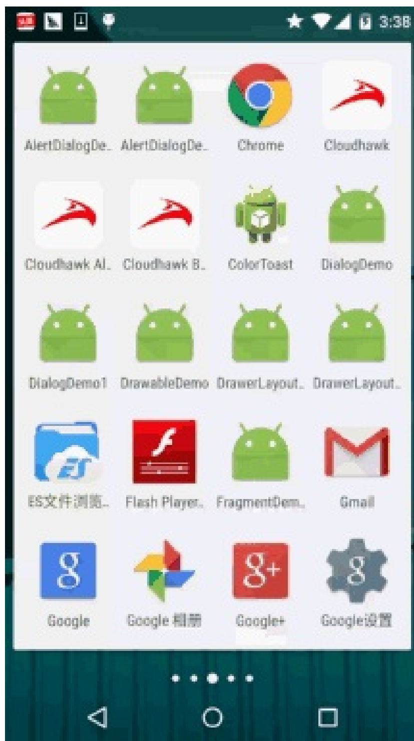
> 用来管理一组Drawable的,我们可以为里面的drawable设置不同的level， 当他们绘制的时候，会根据level属性值获取对应的drawable绘制到画布上，根节点为:**<level-list>**他并没有可以设置的属性，我们能做的只是设置每个**<item>** 的属性！

item可供设置的属性如下：

> - **drawable**:引用的位图资源,如果为空徐璈有一个Drawable类型的子节点
> - **minlevel:level**对应的最小值
> - **maxlevel:level**对应的最大值

使用示例：

运行效果图：



代码实现：

通过shapeDrawable画圆，一式五份，改下宽高即可：

shape_cir1.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#2C96ED"/>
    <size android:height="20dp" android:width="20dp"/>
</shape>
```

接着到LevelListDrawable，这里我们设置五层：

level_cir.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/andro
    <item android:drawable="@drawable/shape_cir1" android:maxLevel=
    <item android:drawable="@drawable/shape_cir2" android:maxLevel=
    <item android:drawable="@drawable/shape_cir3" android:maxLevel=
    <item android:drawable="@drawable/shape_cir4" android:maxLevel=
    <item android:drawable="@drawable/shape_cir5" android:maxLevel=
</level-list>
```

最后MainActivity写如下代码：

```
public class MainActivity extends AppCompatActivity {

    private ImageView img_show;

    private LevelListDrawable ld;
    private Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            if (msg.what == 0x123) {
                if (ld.getLevel() > 10000) ld.setLevel(0);
                img_show.setImageLevel(ld.getLevel() + 2000);
            }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img_show = (ImageView) findViewById(R.id.img_show);
        ld = (LevelListDrawable) img_show.getDrawable();
        img_show.setImageLevel(0);
        new Timer().schedule(new TimerTask() {
            @Override
            public void run() {
                handler.sendEmptyMessage(0x123);
            }
        }, 0, 100);
    }
}
```

也很简单，一个Timer定时器,handler修改level值~

4.StateListDrawable

> 好了终于迎来了最后一个drawable：StateListDrawable，这个名字看上去模式，其实我们以前就用到了，还记得为按钮设置不同状态的drawable的 **<selctor>**吗？没错，用到的 就是这个StateListDrawable！

可供设置的属性如下：

> - **drawable**:引用的Drawable位图,我们可以把他放到最前面,就表示组件的正常状态~
> - **state_focused**:是否获得焦点
> - **state_window_focused**:是否获得窗口焦点
> - **state_enabled**:控件是否可用
> - **state_checkable**:控件可否被勾选,eg:checkbox
> - **state_checked**:控件是否被勾选
> - **state_selected**:控件是否被选择,针对有滚轮的情况
> - **state_pressed**:控件是否被按下
> - **state_active**:控件是否处于活动状态,eg:slidingTab
> - **state_single**:控件包含多个子控件时,确定是否只显示一个子控件
> - **state_first**:控件包含多个子控件时,确定第一个子控件是否处于显示状态
> - **state_middle**:控件包含多个子控件时,确定中间一个子控件是否处于显示状态
> - **state_last**:控件包含多个子控件时,确定最后一个子控件是否处于显示状态

使用示例：

那就来写个简单的圆角按钮吧！

运行效果图：



代码实现：

那就先通过shapeDrawable来画两个圆角矩形，只是颜色不一样而已：

shape_btn_normal.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#DD788A"/>
    <corners android:radius="5dp"/>
    <padding android:top="2dp" android:bottom="2dp"/>
</shape>
```

接着我们来写个selctor：**selctor_btn.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/androic
    <item android:state_pressed="true" android:drawable="@drawable/
    <item android:drawable="@drawable/shape_btn_normal"/>
</selector>
```

然后按钮设置android:background="@drawable/selctor_btn"就可以了~ 你可以根据自己需求改成矩形或者椭圆，圆形等！

本节小结：

> 好的，关于Android中的13种不同类型的Drawable已经讲解完毕了，当然，这只是基础，实际 开发中肯定还有各种高逼格的用法，这就要靠大家去扩展了，这里只是给大家一个引导！
>
> 嗯，时间关系，上述的例子都是一个个试的，所以最后的demo乱七八糟哈，可能 你对这些素材又需要，还是贴下，有需要的自行下载：DrawableDemo.zip 嗯，谢谢~祝周末愉快

8.2.1 Bitmap(位图)全解析 Part 1

本节引言：

在上一节中我们对Android中的13种类型的Drawable的类型进行了讲解，有没有应用到自己的 项目当中呢？而本节我们来探讨的是Bitmap(位图)的一些使用，而在开始本节的内容之前我们 先来区分几个名词的概念：

- **Drawable**：通用的图形对象，用于装载常用格式的图像，既可以是PNG，JPG这样的图像， 也是前面学的那13种Drawable类型的可视化对象！我们可以理解成一个用来放画的——画框！
- **Bitmap(位图)**：我们可以把他看作一个画架，我们先把画放到上面，然后我们可以 进行一些处理，比如获取图像文件信息，做旋转切割，放大缩小等操作！
- **Canvas(画布)**：如其名，画布，我们可以在上面作画(绘制)，你既可以用**Paint(画笔)**， 来画各种形状或者写字，又可以用**Path(路径)**来绘制多个点，然后连接成各种图形！
- **Matrix(矩阵)**：用于图形特效处理的，颜色矩阵(ColorMatrix)，还有使用Matrix进行图像的 平移，缩放，旋转，倾斜等！

而上述的这些都是Android中的底层图形类：**android.graphics**给我们提供的接口！ 嗯，话不多说开始本节内容！ PS：官方文档：Bitmap

1.了解Bitmap，BitmapFactory，BitmapFacotry.Options

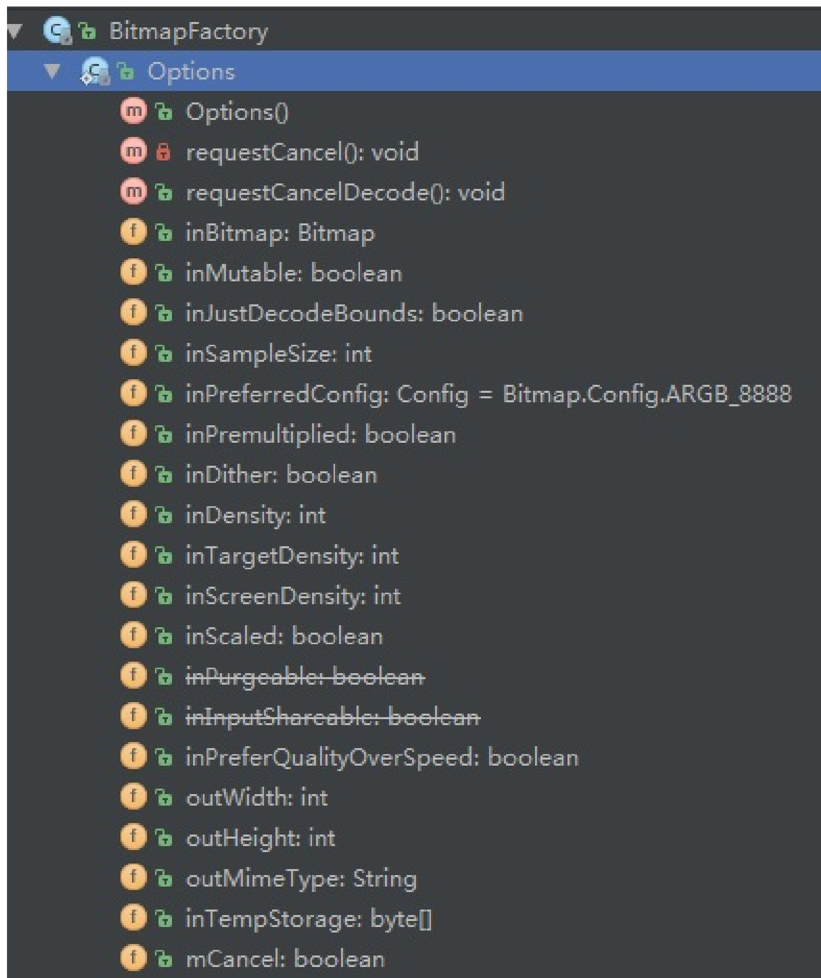
如题，本来可以直接说着三个东东的关系的，但是我就是要傲娇，就要看代码！ 如果你打开Bitmap类的源码，你会看到Bitmap的构造方法上有这样一段东东：

```
/**
 * Private constructor that must received an already allocated native bitmap
 * int (pointer).
 */
// called from JNI
Bitmap(long nativeBitmap, byte[] buffer, int width, int height, int density,
        boolean isMutable, boolean requestPremultiplied,
        byte[] ninePatchChunk, NinePatch.InsetStruct ninePatchInsets) {
    if (nativeBitmap == 0) {
        throw new RuntimeException("internal error: native bitmap is 0");
    }
```

大概想说的就是：Bitmap的构造方法是私有的，外面不能实例化，只能通过JNI实例化！ 当然，肯定也会给我们提供一个接口给我们来创建Bitmap的，而这个接口类就是：**BitmapFactory**！ 来来来，打开BitmapFactory类，我们点下左边的Structure可以看到BitmapFactory给我们 提供了这些方法，大部分都是decodeXxx，通过各种形式来创建Bitmap的！



接着我们又发现了，每一种方法，都会有一个Options类型的参数，点进去看看： 于是乎我们发现了这货是一个静态内部类:**BitmapFacotry.Options**! 而他是用来设置decode时的选项的！

> 
>
>
> 我们对这里的某些参数的值进行设置，比如inJustDecodeBounds设置为true避免OOM(内存溢出)， 什么，不知道OOM，没事，等下一点点跟你说清楚！最后回到我们的Bitmap！嗯，Bitmap中的 方法比较多，就不一一进行讲解了，我们从中挑几个用得较多的来讲解！ 中文文档：Android中文API（136） —— Bitmap

2.Bitmap常用方法

> 普通方法
>
> - public boolean **compress** (Bitmap.CompressFormat format, int quality, OutputStream stream) 将位图的压缩到指定的OutputStream，可以理解成将Bitmap保存到文件中！ **format**：格式，PNG，JPG等； **quality**：压缩质量，0-100，0表示最低画质压缩，100最大质量(PNG无损，会忽略品质设定) **stream**：输出流 返回值代表是否成功压缩到指定流！
> - void **recycle**()：回收位图占用的内存空间，把位图标记为Dead
> - boolean **isRecycled**()：判断位图内存是否已释放
> - int **getWidth**()：获取位图的宽度
> - int **getHeight**()：获取位图的高度
> - boolean **isMutable**()：图片是否可修改
> - int **getScaledWidth**(Canvas canvas)：获取指定密度转换后的图像的宽度
> - int **getScaledHeight**(Canvas canvas)：获取指定密度转换后的图像的高度
>
> 静态方法：
>
> - Bitmap **createBitmap**(Bitmap src)：以src为原图生成不可变得新图像
> - Bitmap **createScaledBitmap**(Bitmap src, int dstWidth,int dstHeight, boolean filter)：以src为原图，创建新的图像，指定新图像的高宽以及是否变。
> - Bitmap **createBitmap**(int width, int height, Config config)：创建指定格式、大小的位图
> - Bitmap **createBitmap**(Bitmap source, int x, int y, int width, int height)以source为原图，创建新的图片，指定起始坐标以及新图像的高宽。
> - public static Bitmap **createBitmap**(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)

BitmapFactory.Option可设置参数：

> - boolean **inJustDecodeBounds**——如果设置为true，不获取图片，不分配内存，但会返回图片的高宽度信息。
> - int **inSampleSize**——图片缩放的倍数。如果设为4，则宽和高都为原来的1/4，则图是原来的1/16。
> - int **outWidth**——获取图片的宽度值
> - int **outHeight**——获取图片的高度值
> - int **inDensity**——用于位图的像素压缩比
> - int **inTargetDensity**——用于目标位图的像素压缩比（要生成的位图）
> - boolean **inScaled**——设置为true时进行图片压缩，从inDensity到inTargetDensity。

好吧，就贴这么多吧，要用自己查文档~

3.获取Bitmap位图

> 从资源中获取位图的方式有两种：通过BitmapDrawable或者BitmapFactory，下面演示下： 我们首先得获得这个

BitmapDrawable方法：

你可以创建一个构造一个BitmapDrawable对象，比如通过流构建BitmapDrawable：

```
BitmapDrawable bmpMeizi = new BitmapDrawable(getAssets().open("pic_
Bitmap mBitmap = bmpMeizi.getBitmap();
img_bg.setImageBitmap(mBitmap);
```

BitmapFactory方法：

都是静态方法，直接调，可以通过资源ID、路径、文件、数据流等方式来获取位图！

```
//通过资源ID
private Bitmap getBitmapFromResource(Resources res, int resId) {
      return BitmapFactory.decodeResource(res, resId);
}

//文件
private Bitmap getBitmapFromFile(String pathName) {
      return BitmapFactory.decodeFile(pathName);
}

//字节数组
public Bitmap Bytes2Bimap(byte[] b) {
    if (b.length != 0) {
        return BitmapFactory.decodeByteArray(b, 0, b.length);
    } else {
        return null;
    }
}

//输入流
private Bitmap getBitmapFromStream(InputStream inputStream) {
      return BitmapFactory.decodeStream(inputStream);
}
```

4.获取Bitmap的相关信息：

这个，只要我们获取了Bitmap对象，就可以调用相关方法来获取对应的参数了，getByteCount获得大小， getHeight和getWidth这些~这里就不写了，自己查文档！

5.抠图片上的某一角下来

有时，可能你想把图片上的某一角扣下来，直接通过Bitmap的createBitmap()扣下来即可 参数依次为：处理的bitmap对象，起始x,y坐标，以及截取的宽高

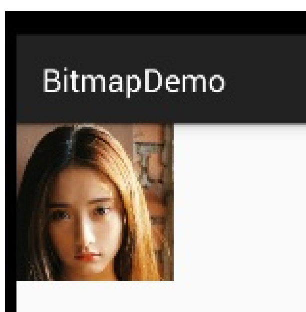
```
Bitmap bitmap1 = BitmapFactory.decodeResource(getResources(), R.mip
Bitmap bitmap2 = Bitmap.createBitmap(bitmap1,100,100,200,200);
img_bg = (ImageView) findViewById(R.id.img_bg);
img_bg.setImageBitmap(bitmap2);
```

运行效果图：

原图：

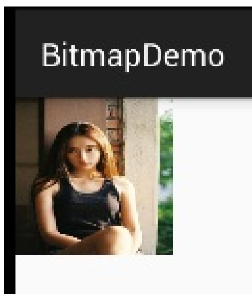


切下来的一角：



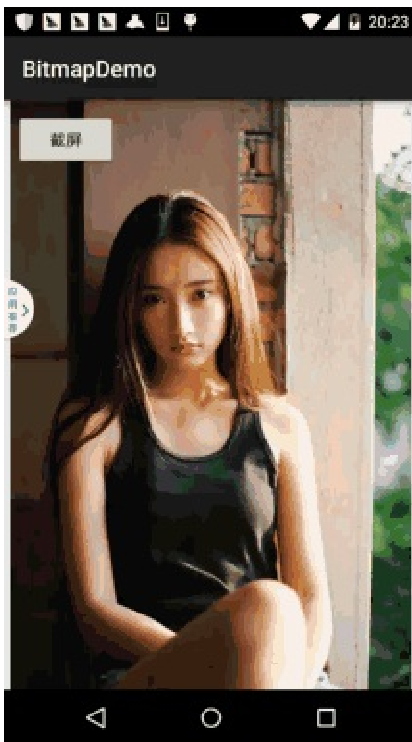
6.对Bitmap进行缩放

我们这里不用Matrix来对Bitmap，而是直接使用Bitmap给我们提供的**createScaledBitmap**来实现， 参数依次是：处理的bitmap对象，缩放后的宽高，



7.使用Bitmap进行截屏

运行效果图：



实现代码：

```
public class MainActivity extends AppCompatActivity {
    static ByteArrayOutputStream byteOut = null;
    private Bitmap bitmap = null;
    private Button btn_cut;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn_cut = (Button) findViewById(R.id.btn_cut);
        btn_cut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                captureScreen();
            }
```

```
        });
    }

    public void captureScreen() {
        Runnable action = new Runnable() {
            @Override
            public void run() {
                final View contentView = getWindow().getDecorView(
                try{
                    Log.e("HEHE",contentView.getHeight()+":"+conten
                    bitmap = Bitmap.createBitmap(contentView.getWic
                            contentView.getHeight(), Bitmap.Config.
                    contentView.draw(new Canvas(bitmap));
                    ByteArrayOutputStream byteOut = new ByteArrayOu
                    bitmap.compress(Bitmap.CompressFormat.JPEG, 10(
                    savePic(bitmap, "sdcard/short.png");
                }catch (Exception e){e.printStackTrace();}
                finally {
                    try{
                        if (null != byteOut)
                            byteOut.close();
                        if (null != bitmap && !bitmap.isRecycled()
//                              bitmap.recycle();
                            bitmap = null;
                        }
                    }catch (IOException e){e.printStackTrace();}

                }
            }
        };
        try {
            action.run();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void savePic(Bitmap b, String strFileName) {
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(strFileName);
            if (null != fos) {
                boolean success= b.compress(Bitmap.CompressFormat.F
                fos.flush();
                fos.close();
                if(success)
                    Toast.makeText(MainActivity.this, "截屏成功", To
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    }
}
```

代码分析：

> 代码非常简单，final View contentView = getWindow().getDecorView();这句代码是获取当前XML 根节点的View！然后设置截屏的大小，调用下 contentView.draw(new Canvas(bitmap));好了，然后 bitmap转换成流，接着写入SD卡，没了~当然从结果我们也可以看出，截图截取的是改APP的内容而已！如果要截全屏，自行谷歌~!

本节小结：

> 本节给大家讲解下Bitmap，BitmapFactory和他的静态内部类Options，以及BitmapDrawable的 基本使用，其实Bitmap我们知道怎么创建就好了，他的扩展一般是通过Matrix和Canvas来实现的， Bitmap，我们更多的时候关注的是OOM问题，下一节我们就来学习下如何避免Bitmap的OOM问题！ 谢谢~
>
> 

8.2.2 Bitmap引起的OOM问题

本节引言：

> 上节，我们已经学习了Bitmap的基本用法，而本节我们要来探讨的Bitmap的OOM问题， 大家在实际开发中可能遇到过，或者没遇到过因为Bitmap引起的OOM问题，本节我们 就来围绕这个话题来进行学习~了解什么是OOM，为什么会引起OOM，改善因Bitmap引起的 OOM问题~

1.什么是OOM？为什么会引起OOM？

答：**Out Of Memory**(内存溢出)，我们都知道Android系统会为每个APP分配一个独立的工作空间， 或者说分配一个单独的Dalvik虚拟机，这样每个APP都可以独立运行而不相互影响！而Android对于每个 Dalvik虚拟机都会有一个最大内存限制，如果当前占用的内存加上我们申请的内存资源超过了这个限制 ，系统就会抛出OOM错误！另外，这里别和RAM混淆了，即时当前RAM中剩余的内存有1G多，但是OOM还是会发生！别把RAM(物理内存)和OOM扯到一起！另外RAM不足的话，就是杀应用了，而不是仅仅是OOM了！ 而这个Dalvik中的最大内存标准，不同的机型是不一样的，可以调用：

```
ActivityManager activityManager =  (ActivityManager)context.getSyst
```

获得正常的最大内存标准，又或者直接在命令行键入：

```
adb shell getprop | grep dalvik.vm.heapgrowthlimit
```

你也可以打开系统源码/system/build.prop文件，看下文件中这一部分的信息得出：

```
dalvik.vm.heapstartsize=8m dalvik.vm.heapgrowthlimit=192m dalvik.vm
```

我们关注的地方有三个：heapstartsize堆内存的初始大小，heapgrowthlimit标准的应用的最大堆 内存大小，heapsize则是设置了使用android:largeHeap的应用的最大堆内存大小！

我这里试了下手头几个机型的正常最大内存分配标准：

```
C:\Users\    >adb shell getprop | grep dalvik.vm.heapgrowthlimit
[dalvik.vm.heapgrowthlimit]: [192m]          真机 Nexus 5

C:\Users\    >adb shell getprop | grep dalvik.vm.heapgrowthlimit
[dalvik.vm.heapgrowthlimit]: [96m]           模拟器Nexus 4

C:\Users\    >adb shell getprop | grep dalvik.vm.heapgrowthlimit
[dalvik.vm.heapgrowthlimit]: [64m]           模拟器Nexus S

C:\Users\    >adb shell getprop | grep dalvik.vm.heapgrowthlimit
error: more than one device and emulator

C:\Users\    >adb shell getprop | grep dalvik.vm.heapgrowthlimit
[dalvik.vm.heapgrowthlimit]: [48m]           真机Nexus S
```

你也可以试试自己手头的机子~

好啦，不扯了，关于OOM问题的产生，就扯到这里，再扯就到内存管理那一块了，可是个大块头，现在还啃不动...下面我们来看下避免Bitmap OOM的一些技巧吧！

2.避免Bitmap引起的OOM技巧小结

1）采用低内存占用量的编码方式

上一节说了**BitmapFactory.Options**这个类，我们可以设置下其中的inPreferredConfig属性，默认是**Bitmap.Config.ARGB_8888**，我们可以修改成**Bitmap.Config.ARGB_4444** Bitmap.Config ARGB_4444：每个像素占四位，即A=4，R=4，G=4，B=4，那么一个像素点占4+4+4+4=16位 Bitmap.Config ARGB_8888：每个像素占八位，即A=8，R=8，G=8，B=8，那么一个像素点占8+8+8+8=32位 默认使用ARGB_8888，即一个像素占4个字节！

2）图片压缩

同样是BitmapFactory.Options，我们通过**inSampleSize**设置缩放倍数，比如写2，即长宽变为原来的1/2，图片就是原来的1/4，如果不进行缩放的话设置为1即可！但是不能一味的压缩，毕竟这个值太小 的话，图片会很模糊，而且要避免图片的拉伸变形，所以需要我们在程序中动态的计算，这个 inSampleSize的合适值，而Options中又有这样一个方法：**inJustDecodeBounds**，将该参数设置为 true后，decodeFiel并不会分配内存空间，但是可以计算出原始图片的长宽，调用options.**outWidth/outHeight**获取出图片的宽高，然后通过一定的算法，即可得到适合的 inSampleSize，这里感谢街神提供的代码——摘自鸿洋blog！

```
public  static  int caculateInSampleSize(BitmapFactory.Options opt
```

然后使用下上述的方法即可：

```
BitmapFactory.Options options =  new  BitmapFactory.Options(); opti
```

3.及时回收图像

如果引用了大量的Bitmap对象，而应用又不需要同时显示所有图片。可以将暂时不用到的Bitmap对象 及时回收掉。对于一些明确知道图片使用情况的场景可以主动recycle回收，比如引导页的图片，使用 完就recycle，帧动画，加载一张，画一张，释放一张！使用时加载，不显示时直接置null或recycle！ 比如：imageView.setImageResource(0); 不过某些情况下会出现特定图片反复加载，释放，再加载等，低效率的事情...

4.其他方法

下面这些方法，我并没有用过，大家可以自行查阅相关资料：

1.简单通过SoftReference引用方式管理图片资源

建个SoftReference的hashmap 使用图片时先查询这个hashmap是否有softreference， softreference里的图片是否为空， 如果为空就加载图片到softreference并加入hashmap。 无需再代码里显式的处理图片的回收与释放，gc会自动处理资源的释放。 这种方式处理起来简单实用，能一定程度上避免前一种方法反复加载释放的低效率。但还不够优化。

示例代码：

```
private  Map<String,  SoftReference<Bitmap>> imageMap =  new  HashM
```

2.LruCache + sd的缓存方式

> Android 3.1版本起，官方还提供了LruCache来进行cache处理，当存储Image的大小大于LruCache 设定的值，那么近期使用次数最少的图片就会被回收掉，系统会自动释放内存！

使用示例：

步骤：

1）要先设置缓存图片的内存大小，我这里设置为手机内存的1/8, 手机内存的获取方式：int MAXMEMONRY = (int) (Runtime.getRuntime() .maxMemory() / 1024);

2）LruCache里面的键值对分别是URL和对应的图片

3）重写了一个叫做sizeOf的方法，返回的是图片数量。

```
private  LruCache<String,  Bitmap> mMemoryCache;  private  LruCache
```

4）下面的方法分别是清空缓存、添加图片到缓存、从缓存中取得图片、从缓存中移除。

移除和清除缓存是必须要做的事，因为图片缓存处理不当就会报内存溢出，所以一定要引起注意。

```
public  void clearCache()  {  if  (mMemoryCache !=  null)  {  if  (
  * 移除缓存
  *
  * @param key
  */  public  synchronized  void removeImageCache(String key)  {  if
```

上述内容摘自——图片缓存之内存缓存技术LruCache,软引用

本节小结：

> 本节给大家讲解了OOM问题的发生缘由，也总结了一下网上给出的一些避免因Bitmap而引起OOM 的一些方案，因为公司做的APP都是地图类的，很少涉及到图片，所以笔者并没有遇到过OOM的问题，所以对此并不怎么熟悉~后续在进阶课程的内存管理，我们再慢慢纠结这个OOM的问题，好的，本节就到这里，谢谢~

参考文献：Android应用中OOM问题剖析和解决方案

8.3.1 三个绘图工具类详解

本节引言：

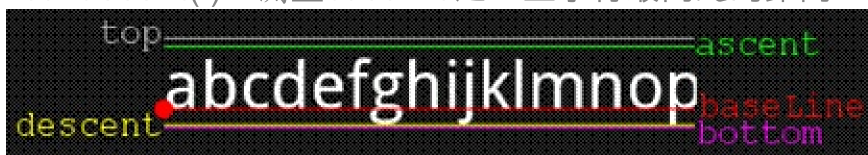
> 上两小节我们学习了Drawable以及Bitmap，都是加载好图片的，而本节我们要学习的绘图相关的 一些API，他们分别是Canvas(画布)，Paint(画笔)，Path(路径)！本节非常重要，同时也是我们 自定义View的基础哦~好的，话不多说开始本节内容~
>
> 官方API文档：Canvas；Paint；Path；

1.相关方法详解

1)Paint(画笔):

> 就是画笔,用于设置绘制风格,如:线宽(笔触粗细),颜色,透明度和填充风格等 直接使用无参构造方法就可以创建Paint实例: **Paint paint = new Paint( );**
>
> 我们可以通过下述方法来设置Paint(画笔)的相关属性,另外,关于这个属性有两种, 图形绘制相关与文本绘制相关:
>
> - **setARGB**(int a,int r,int g,int b): 设置绘制的颜色，a代表透明度，r，g，b代表颜色值。
> - **setAlpha**(int a): 设置绘制图形的透明度。
> - **setColor**(int color): 设置绘制的颜色，使用颜色值来表示，该颜色值包括透明度和RGB颜色。
> - **setAntiAlias**(boolean aa): 设置是否使用抗锯齿功能，会消耗较大资源，绘制图形速度会变慢。
> - **setDither**(boolean dither): 设定是否使用图像抖动处理，会使绘制出来的图片颜色更加平滑和饱满，图像更加清晰
> - **setFilterBitmap**(boolean filter)： 如果该项设置为true，则图像在动画进行中会滤掉对Bitmap图像的优化操作， 加快显示速度，本设置项依赖于dither和xfermode的设置
> - **setMaskFilter**(MaskFilter maskfilter)： 设置MaskFilter，可以用不同的MaskFilter实现滤镜的效果，如滤化，立体等
> - **setColorFilter**(ColorFilter colorfilter)： 设置颜色过滤器，可以在绘制颜色时实现不用颜色的变换效果
> - **setPathEffect**(PathEffect effect) 设置绘制路径的效果，如点画线等
> - **setShader**(Shader shader)： 设置图像效果，使用Shader可以绘制出各种渐变效果
> - **setShadowLayer**(float radius ,float dx,float dy,int color)：在图形下面设置阴影层， 产生阴影效果， radius为阴影的角度，dx和dy为阴影在x轴和y轴上的距离，color为阴影的颜色
> - **setStyle**(Paint.Style style)： 设置画笔的样式， 为FILL，

FILL_OR_STROKE，或STROKE
- **setStrokeCap**(Paint.Cap cap)： 当画笔样式为STROKE或FILL_OR_STROKE时，设置笔刷的图形样式， 如圆形样Cap.ROUND,或方形样式Cap.SQUARE
- **setSrokeJoin**(Paint.Join join)： 设置绘制时各图形的结合方式，如平滑效果等
- **setStrokeWidth**(float width)： 当画笔样式为STROKE或FILL_OR_STROKE时，设置笔刷的粗细度
- **setXfermode**(Xfermode xfermode)： 设置图形重叠时的处理方式，如合并，取交集或并集，经常用来制作橡皮的擦除效果
- **setFakeBoldText**(boolean fakeBoldText)： 模拟实现粗体文字，设置在小字体上效果会非常差
- **setSubpixelText**(boolean subpixelText)： 设置该项为true，将有助于文本在LCD屏幕上的显示效果
- **setTextAlign**(Paint.Align align)： 设置绘制文字的对齐方向
- **setTextScaleX**(float scaleX)： 设置绘制文字x轴的缩放比例，可以实现文字的拉伸的效果
- **setTextSize**(float textSize)： 设置绘制文字的字号大小
- **setTextSkewX**(float skewX)： 设置斜体文字，skewX为倾斜弧度
- **setTypeface**(Typeface typeface)： 设置Typeface对象，即字体风格，包括粗体，斜体以及衬线体，非衬线体等
- **setUnderlineText**(boolean underlineText)： 设置带有下划线的文字效果
- **setStrikeThruText**(boolean strikeThruText)： 设置带有删除线的效果
- **setStrokeJoin**(Paint.Join join)： 设置结合处的样子，Miter:结合处为锐角， Round:结合处为圆弧：BEVEL：结合处为直线
- **setStrokeMiter**(float miter)：设置画笔倾斜度
- **setStrokeCap** (Paint.Cap cap)：设置转弯处的风格 其他常用方法：
- float **ascent**( )：测量baseline之上至字符最高处的距离



- float **descent**()：baseline之下至字符最低处的距离
- int **breakText**(char[] text, int index, int count, float maxWidth, float[] measuredWidth)： 检测一行显示多少文字
- **clearShadowLayer**( )：清除阴影层 其他的自行查阅文档~

2)Canvas(画布):

画笔有了，接着就到画笔(Canvas)，总不能凭空作画是吧~常用方法如下：

首先是构造方法，Canvas的构造方法有两种：

Canvas(): 创建一个空的画布，可以使用setBitmap()方法来设置绘制具体的画布。

Canvas(Bitmap bitmap): 以bitmap对象创建一个画布，将内容都绘制在bitmap上，因此bitmap不得为null。

接着是 **1.drawXXX()**方法族：以一定的坐标值在当前画图区域画图， 另外图层会叠加， 即后面绘画的图层会覆盖前面绘画的图层。 比如：

- **drawRect**(RectF rect, Paint paint) ：绘制区域，参数一为RectF一个区域
- **drawPath**(Path path, Paint paint) ：绘制一个路径，参数一为Path路径对象
- **drawBitmap**(Bitmap bitmap, Rect src, Rect dst, Paint paint) ： 贴图，参数一就是我们常规的Bitmap对象，参数二是源区域(这里是bitmap)， 参数三是目标区域(应该在canvas的位置和大小)，参数四是Paint画刷对象， 因为用到了缩放和拉伸的可能，当原始Rect不等于目标Rect时性能将会有大幅损失。
- **drawLine**(float startX, float startY, float stopX, float stopY, Paintpaint) ：画线，参数一起始点的x轴位置，参数二起始点的y轴位置，参数三终点的x轴水平位置， 参数四y轴垂直位置，最后一个参数为Paint 画刷对象。
- **drawPoint**(float x, float y, Paint paint)： 画点，参数一水平x轴，参数二垂直y轴，第三个参数为Paint对象。
- **drawText**(String text, float x, floaty, Paint paint) ： 渲染文本，Canvas类除了上面的还可以描绘文字，参数一是String类型的文本， 参数二x轴，参数三y轴，参数四是Paint对象。
- **drawOval**(RectF oval, Paint paint)：画椭圆，参数一是扫描区域，参数二为paint对象；
- **drawCircle**(float cx, float cy, float radius,Paint paint)： 绘制圆，参数一是中心点的x轴，参数二是中心点的y轴，参数三是半径，参数四是paint对象；
- **drawArc**(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)： 画弧，参数一是RectF对象，一个矩形区域椭圆形的界限用于定义在形状、大小、电弧，参数二是起始角 (度)在电弧的开始，参数三扫描角(度)开始顺时针测量的，参数四是如果这是真的话,包括椭圆中心的电 弧,并关闭它,如果它是假这将是一个弧线,参数五是Paint对象；

2.**clipXXX()**方法族:在当前的画图区域裁剪(clip)出一个新的画图区域，这个画图区域就是canvas 对象的当前画图区域了。比如：clipRect(new Rect())，那么该矩形区域就是canvas的当前画图区域

3.**save()**和**restore()**方法： **save**( )：用来保存Canvas的状态。save之后，可以调用Canvas的平移、放缩、旋转、错切、裁剪等操作！ **restore**（）：用来恢复Canvas之前保存的状态。防止save后对Canvas执行的操作对后续的绘制有影响。 save()和restore()要配对使用(restore可以比save少,但不能多)，若restore调用次数比save多,会报错！

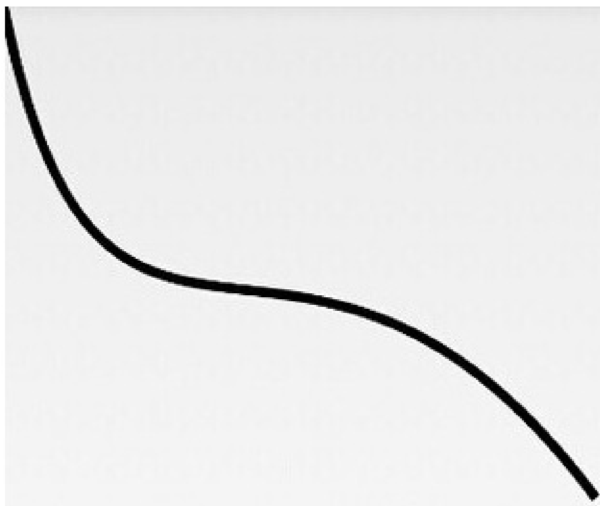
4.**translate**(float dx, float dy)： 平移，将画布的坐标原点向左右方向移动x，向上下方向移动y.canvas的默认位置是在（0,0）

5.**scale**(float sx, float sy)：扩大，x为水平方向的放大倍数，y为竖直方向的放大倍数

6.**rotate**(float degrees)：旋转，angle指旋转的角度，顺时针旋转

3)Path(路径)

> 简单点说就是描点，连线~在创建好我们的Path路径后，可以调用Canvas的 **drawPath**(path,paint) 将图形绘制出来~常用方法如下：
>
> - **addArc**(RectF oval, float startAngle, float sweepAngle：为路径添加一个多边形
> - **addCircle**(float x, float y, float radius, Path.Direction dir)：给path添加圆圈
> - **addOval**(RectF oval, Path.Direction dir)：添加椭圆形
> - **addRect**(RectF rect, Path.Direction dir)：添加一个区域
> - **addRoundRect**(RectF rect, float[] radii, Path.Direction dir)：添加一个圆角区域
> - **isEmpty**()：判断路径是否为空
> - **transform**(Matrix matrix)：应用矩阵变换
> - **transform**(Matrix matrix, Path dst)：应用矩阵变换并将结果放到新的路径中，即第二个参数。
>
> 更高级的效果可以使用PathEffect类！
>
> 几个To：
>
> - **moveTo**(float x, float y)：不会进行绘制，只用于移动移动画笔
> - **lineTo**(float x, float y)：用于直线绘制，默认从(0，0)开始绘制，用moveTo移动！ 比如 mPath.lineTo(300, 300); canvas.drawPath(mPath, mPaint);
> - **quadTo**(float x1, float y1, float x2, float y2)： 用于绘制圆滑曲线，即贝塞尔曲线，同样可以结合moveTo使用！
>
> 
>
> - **rCubicTo**(float x1, float y1, float x2, float y2, float x3, float y3) 同样是用来实现贝塞尔曲线的。 (x1,y1) 为控制点，(x2,y2)为控制点，(x3,y3) 为结束点。 Same as cubicTo, but the coordinates are considered relative to the current point on this contour.就是多一个控制点而已~ 绘制上述的曲线：mPath.moveTo(100, 500); mPath.cubicTo(100, 500, 300, 100, 600, 500); 如果不加上面的那个moveTo的话：则以(0,0)为起点，(100,500)和(300,100)为控制点绘制贝塞尔曲线

> 
>
> - **arcTo**(RectF oval, float startAngle, float sweepAngle)： 绘制弧线（实际是截取圆或椭圆的一部分）ovalRectF为椭圆的矩形，startAngle 为开始角度， sweepAngle 为结束角度。

2.动手试试：

> 属性那么多，肯定要手把手的撸一下，才能加深我们的映像是吧~ 嘿嘿，画图要么在View上画，要么在SurfaceView上画，这里我们在View上画吧， 我们定义一个View类，然后再onDraw()里完成绘制工作！

```
/**
 * Created by Jay on 2015/10/15 0015.
 */
public class MyView extends View{

    private Paint mPaint;

    public MyView(Context context) {
        super(context);
        init();
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init(){
        mPaint = new Paint();
        mPaint.setAntiAlias(true);          //抗锯齿
        mPaint.setColor(getResources().getColor(R.color.puple));//i
        mPaint.setStyle(Paint.Style.FILL);  //画笔风格
        mPaint.setTextSize(36);             //绘制文字大小，单位px
        mPaint.setStrokeWidth(5);           //画笔粗细
    }

    //重写该方法，在这里绘图
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
    }

}
```

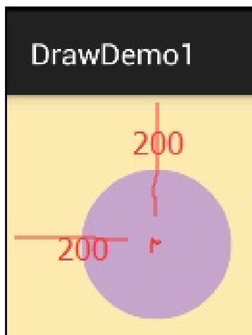
然后布局那里设置下这个View就好，下述代码都写在onDrawable中~

1）设置画布颜色：

```
canvas.drawColor(getResources().getColor(R.color.yellow));   //设置
```

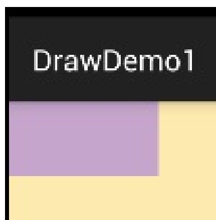
2）绘制圆形：

```
canvas.drawCircle(200, 200, 100, mPaint);           //画实心圆
```



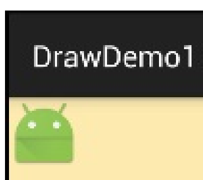
3）绘制矩形：

```
canvas.drawRect(0, 0, 200, 100, mPaint);            //画矩形
```



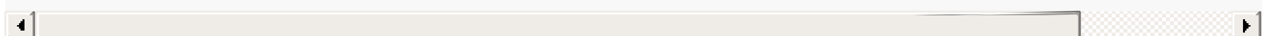
4）绘制Bitmap：

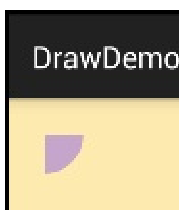
```
canvas.drawBitmap(BitmapFactory.decodeResource(getResources(), R.m
```



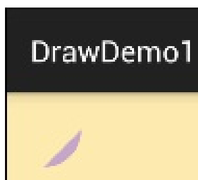
5）绘制弧形区域：

```
canvas.drawArc(new RectF(0, 0, 100, 100),0,90,true,mPaint);  //绘制
```



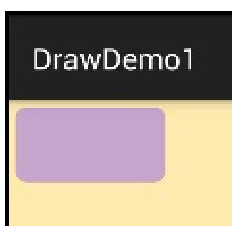


假如true改为false：



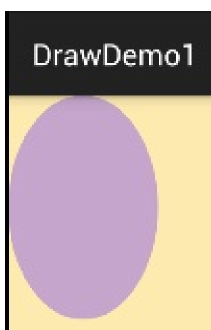
6）绘制圆角矩形

```
canvas.drawRoundRect(new RectF(10,10,210,110),15,15,mPaint); //画圆
```



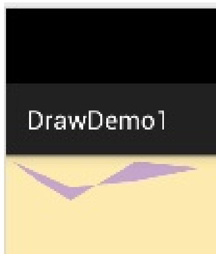
7）绘制椭圆

```
canvas.drawOval(new RectF(0,0,200,300),mPaint); //画椭圆
```



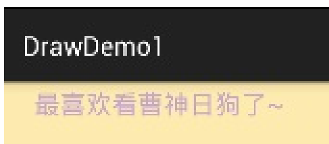
8)绘制多边形：

```
Path path = new Path();
path.moveTo(10, 10); //移动到 坐标10,10
path.lineTo(100, 50);
path.lineTo(200,40);
path.lineTo(300, 20);
path.lineTo(200, 10);
path.lineTo(100, 70);
path.lineTo(50, 40);
path.close();
canvas.drawPath(path,mPaint);
```



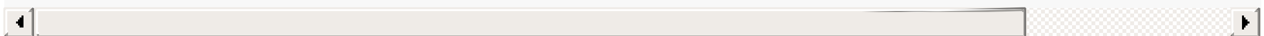
9)绘制文字：

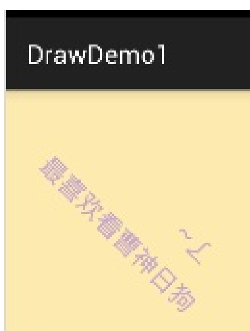
```
canvas.drawText("最喜欢看曹神日狗了~",50,50,mPaint);    //绘制文字
```



你也可以沿着某条Path来绘制这些文字：

```
Path path = new Path();
path.moveTo(50,50);
path.lineTo(100, 100);
path.lineTo(200, 200);
path.lineTo(300, 300);
path.close();
canvas.drawTextOnPath("最喜欢看曹神日狗了~", path, 50, 50, mPaint);
```





10)绘制自定义的图形：

代码来源于网上：

```
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.translate(canvas.getWidth()/2, 200); //将位置移动画纸的坐标
    canvas.drawCircle(0, 0, 100, mPaint); //画圆圈

    //使用path绘制路径文字
    canvas.save();
    canvas.translate(-75, -75);
    Path path = new Path();
    path.addArc(new RectF(0,0,150,150), -180, 180);
    Paint citePaint = new Paint(mPaint);
    citePaint.setTextSize(14);
    citePaint.setStrokeWidth(1);
    canvas.drawTextOnPath("绘制表盘~", path, 28, 0, citePaint);
    canvas.restore();

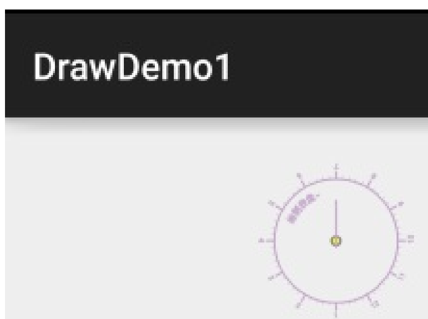
    Paint tmpPaint = new Paint(mPaint); //小刻度画笔对象
    tmpPaint.setStrokeWidth(1);

    float  y=100;
    int count = 60; //总刻度数

    for(int i=0 ; i <count ; i++){
        if(i%5 == 0){
            canvas.drawLine(0f, y, 0, y+12f, mPaint);
            canvas.drawText(String.valueOf(i/5+1), -4f, y+25f, tmpP

        }else{
            canvas.drawLine(0f, y, 0f, y +5f, tmpPaint);
        }
        canvas.rotate(360/count,0f,0f); //旋转画纸
    }

    //绘制指针
    tmpPaint.setColor(Color.GRAY);
    tmpPaint.setStrokeWidth(4);
    canvas.drawCircle(0, 0, 7, tmpPaint);
    tmpPaint.setStyle(Paint.Style.FILL);
    tmpPaint.setColor(Color.YELLOW);
    canvas.drawCircle(0, 0, 5, tmpPaint);
    canvas.drawLine(0, 10, 0, -65, mPaint);
}
```



本节小结：

> 本节我们对android.graphics接口类下的三个绘图API：Canvas(画布)，Paint(画笔)，Path(路径)进行 了学习，方法有很多，别去死记，用到的时候查就好，这里我们先有个大概映像即可，自定义控件那里 我们再来慢慢纠结~好的，就说这么多，谢谢~

8.3.2 绘图类实战示例

本节引言：

> 前两节我们学了Bitmap和一些基本的绘图API的属性以及常用的方法，但心里总觉得有点 不踏实，总得写点什么加深下映像是吧，嗯，本节我们就来写两个简单的例子：
>
> - 1.简单画图板的实现
> - 2.帮美女擦衣服的简单实现
>
> 嘿嘿，第二个例子是小猪刚学安卓写的一个小Demo~嘿嘿~ 开始本节内容~

1.实战示例1：简单画图板的实现：

> 这个相信大家都不陌生，很多手机都会自带一个给用户涂鸦的画图板，这里我们就来写个简单的 例子，首先我们分析下，实现这个东东的一些逻辑：
>
> **Q1**：这个画板放在哪里？
>
> 答：View里，我们自定义一个View，在onDraw()里完成绘制， 另外View还有个onTouchEvent的方法， 我们可以在获取用户的手势操作！
>
> **Q2.**需要准备些什么？
>
> 答：一只画笔(Paint)，一块画布(Canvas)，一个路径(Path)记录用户绘制路线； 另外划线的时候，每次都是从上次拖动时间的发生点到本次拖动时间的发生点！那么之前绘制的 就会丢失，为了保存之前绘制的内容，我们可以引入所谓的"双缓冲"技术： 其实就是每次不是直接绘制到Canvas上，而是先绘制到Bitmap上，等Bitmap上的绘制完了， 再一次性地绘制到View上而已！
>
> **Q3.**具体的实现流程？
>
> 答：初始化画笔，设置颜色等等一些参数；在View的onMeasure()方法中创建一个View大小的Bitmap， 同时创建一个Canvas；onTouchEvent中获得X,Y坐标，做绘制连线，最后invalidate()重绘，即调用 onDraw方法将bitmap的东东画到Canvas上！

好了，逻辑知道了，下面就上代码了：

MyView.java：

```
/**
 * Created by Jay on 2015/10/15 0015.
 */
public class MyView extends View{
```

```
    private Paint mPaint;  //绘制线条的Path
    private Path mPath;      //记录用户绘制的Path
    private Canvas mCanvas;  //内存中创建的Canvas
    private Bitmap mBitmap;  //缓存绘制的内容

    private int mLastX;
    private int mLastY;

    public MyView(Context context) {
        super(context);
        init();
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init(){
        mPath = new Path();
        mPaint = new Paint();   //初始化画笔
        mPaint.setColor(Color.GREEN);
        mPaint.setAntiAlias(true);
        mPaint.setDither(true);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeJoin(Paint.Join.ROUND); //结合处为圆角
        mPaint.setStrokeCap(Paint.Cap.ROUND); // 设置转弯处为圆角
        mPaint.setStrokeWidth(20);   // 设置画笔宽度
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasur
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        int width = getMeasuredWidth();
        int height = getMeasuredHeight();
        // 初始化bitmap,Canvas
        mBitmap = Bitmap.createBitmap(width, height, Bitmap.Config.
        mCanvas = new Canvas(mBitmap);
    }

    //重写该方法，在这里绘图
    @Override
    protected void onDraw(Canvas canvas) {
        drawPath();
        canvas.drawBitmap(mBitmap, 0, 0, null);
    }

    //绘制线条
```

```
    private void drawPath(){
        mCanvas.drawPath(mPath, mPaint);
    }

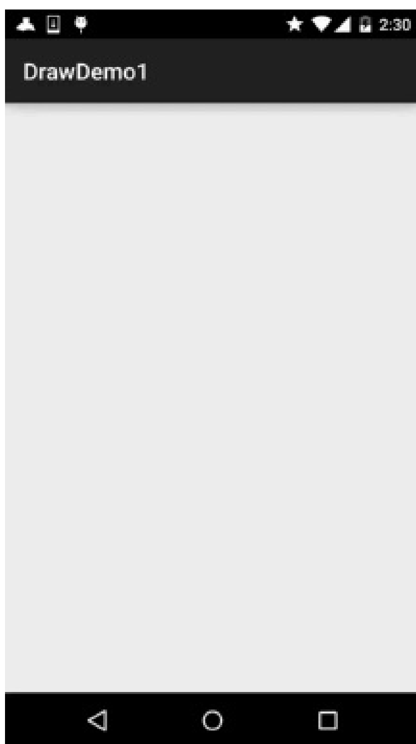
    @Override
    public boolean onTouchEvent(MotionEvent event) {

        int action = event.getAction();
        int x = (int) event.getX();
        int y = (int) event.getY();

        switch (action)
        {
            case MotionEvent.ACTION_DOWN:
                mLastX = x;
                mLastY = y;
                mPath.moveTo(mLastX, mLastY);
                break;
            case MotionEvent.ACTION_MOVE:
                int dx = Math.abs(x - mLastX);
                int dy = Math.abs(y - mLastY);
                if (dx > 3 || dy > 3)
                    mPath.lineTo(x, y);
                mLastX = x;
                mLastY = y;
                break;
        }

        invalidate();
        return true;
    }
}
```

运行效果图：

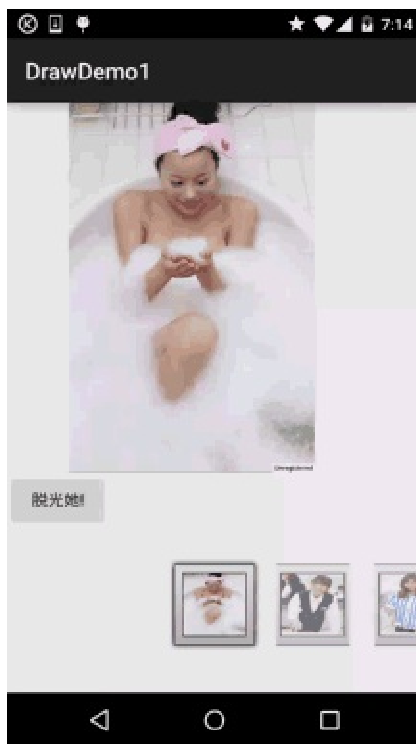


你可以根据自己的需求进行扩展，比如加上修改画笔大小，修改画笔颜色，保存自己画的图等！ 发散思维，自己动手~

2.实战示例2：擦掉美女衣服的实现

> 核心思路是： 利用帧布局，前后两个ImageView，前面的显示未擦掉衣服的情况，后面的显示擦掉衣服后的情况！
>
> 为两个ImageView设置美女图片后，接着为前面的ImageView设置OnTouchListener！在这里对手指 触碰点附近的20*20个像素点，设置为透明！

运行效果图：



代码实现：

Step 1：第一个选妹子的Activity相关的编写，首先是界面，一个ImageView，Button和Gallery！

activity_main.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/img_choose"
        android:layout_width="320dp"
        android:layout_height="320dp" />

    <Button
        android:id="@+id/btn_choose"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="脱光她!" />

    <Gallery
        android:id="@+id/gay_choose"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:spacing="1pt"
        android:unselectedAlpha="0.6" />

</LinearLayout>
```

接着是我们Gallery的Adapter类，这里我们重写下BaseAdapter，而里面就显示一个图片比较简单，就不另外写一个布局了！

MeiziAdapter.java:

```
/**
 * Created by Jay on 2015/10/16 0016.
 */
public class MeiziAdapter extends BaseAdapter{

    private Context mContext;
    private int[] mData;

    public MeiziAdapter() {
    }

    public MeiziAdapter(Context mContext,int[] mData) {
        this.mContext = mContext;
        this.mData = mData;
    }

    @Override
    public int getCount() {
        return mData.length;
    }

    @Override
    public Object getItem(int position) {
        return mData[position];
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup p
        ImageView imgMezi = new ImageView(mContext);
        imgMezi.setImageResource(mData[position]);         //创建一
        imgMezi.setScaleType(ImageView.ScaleType.FIT_XY);      //设
        imgMezi.setLayoutParams(new Gallery.LayoutParams(250, 250)
        TypedArray typedArray = mContext.obtainStyledAttributes(R.s
        imgMezi.setBackgroundResource(typedArray.getResourceId(R.st
        return imgMezi;
    }
}
```

最后到我们的Activity，也很简单，无非是为gallery设置onSelected事件，点击按钮后把，当前选中的 Position传递给下一个页面！

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Adap
        View.OnClickListener {
```

```
    private Context mContext;
    private ImageView img_choose;
    private Button btn_choose;
    private Gallery gay_choose;
    private int index = 0;
    private MeiziAdapter mAdapter = null;
    private int[] imageIds = new int[]
            {
                    R.mipmap.pre1, R.mipmap.pre2, R.mipmap.pre3, R.
                    R.mipmap.pre5, R.mipmap.pre6, R.mipmap.pre7, R.
                    R.mipmap.pre9, R.mipmap.pre10, R.mipmap.pre11,
                    R.mipmap.pre13, R.mipmap.pre14, R.mipmap.pre15,
                    R.mipmap.pre17, R.mipmap.pre18, R.mipmap.pre19,
                    R.mipmap.pre21
            };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        bindViews();
    }

    private void bindViews() {
        img_choose = (ImageView) findViewById(R.id.img_choose);
        btn_choose = (Button) findViewById(R.id.btn_choose);
        gay_choose = (Gallery) findViewById(R.id.gay_choose);

        mAdapter = new MeiziAdapter(mContext, imageIds);
        gay_choose.setAdapter(mAdapter);
        gay_choose.setOnItemSelectedListener(this);
        btn_choose.setOnClickListener(this);

    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, ir
        img_choose.setImageResource(imageIds[position]);
        index = position;
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }

    @Override
    public void onClick(View v) {
        Intent it = new Intent(mContext,CaClothes.class);
        Bundle bundle = new Bundle();
        bundle.putCharSequence("num", Integer.toString(index));
        it.putExtras(bundle);
```

```
        startActivity(it);
    }
}
```

接着是我们擦掉妹子衣服的页面了，布局比较简单，FrameLayout + 前后两个ImageView：

activity_caclothes.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/andr
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/img_after"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <ImageView
        android:id="@+id/img_before"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</FrameLayout>
```

接着到就到Java部分的代码了：

CaClothes.java：

```
/**
 * Created by Jay on 2015/10/16 0016.
 */
public class CaClothes extends AppCompatActivity implements View.On

    private ImageView img_after;
    private ImageView img_before;
    private Bitmap alterBitmap;
    private Canvas canvas;
    private Paint paint;
    private Bitmap after;
    private Bitmap before;
    private int position;

    int[] imageIds1 = new int[]
            {
                    R.mipmap.pre1, R.mipmap.pre2, R.mipmap.pre3, R
                    R.mipmap.pre5, R.mipmap.pre6, R.mipmap.pre7, R
```

```
                    R.mipmap.pre9, R.mipmap.pre10, R.mipmap.pre11,
                    R.mipmap.pre13, R.mipmap.pre14, R.mipmap.pre15,
                    R.mipmap.pre17, R.mipmap.pre18, R.mipmap.pre19,
                    R.mipmap.pre21
            };

    int[] imageIds2 = new int[]
            {
                    R.mipmap.after1, R.mipmap.after2, R.mipmap.afte
                    R.mipmap.after5, R.mipmap.after6, R.mipmap.afte
                    R.mipmap.after9, R.mipmap.after10, R.mipmap.aft
                    R.mipmap.after13, R.mipmap.after14, R.mipmap.af
                    R.mipmap.after17, R.mipmap.after18, R.mipmap.af
                    R.mipmap.after21
            };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_caclothes);

        Bundle bd = getIntent().getExtras();
        position = Integer.parseInt(bd.getString("num"));
        bindViews();

    }

    private void bindViews() {
        img_after = (ImageView) findViewById(R.id.img_after);
        img_before = (ImageView) findViewById(R.id.img_before);

        BitmapFactory.Options opts = new BitmapFactory.Options();
        opts.inSampleSize = 1;
        after = BitmapFactory.decodeResource(getResources(), imageI
        before = BitmapFactory.decodeResource(getResources(), image
        //定义出来的是只读图片

        alterBitmap = Bitmap.createBitmap(before.getWidth(), before
        canvas = new Canvas(alterBitmap);
        paint = new Paint();
        paint.setStrokeCap(Paint.Cap.ROUND);
        paint.setStrokeJoin(Paint.Join.ROUND);
        paint.setStrokeWidth(5);
        paint.setColor(Color.BLACK);
        paint.setAntiAlias(true);
        canvas.drawBitmap(before, new Matrix(), paint);
        img_after.setImageBitmap(after);
        img_before.setImageBitmap(before);
        img_before.setOnTouchListener(this);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
```

```
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    break;
                case MotionEvent.ACTION_MOVE:
                    int newX = (int) event.getX();
                    int newY = (int) event.getY();
                    //setPixel方法是将某一个像素点设置成一个颜色，而这里我们把
                    //另外通过嵌套for循环将手指触摸区域的20*20个像素点设置为透
                    for (int i = -20; i < 20; i++) {
                        for (int j = -20; j < 20; j++) {
                            if (i + newX >= 0 && j + newY >= 0 && i + n
                                alterBitmap.setPixel(i + newX, j + new
                            }
                        }
                    img_before.setImageBitmap(alterBitmap);
                    break;
            }
            return true;
        }
    }
```

代码也不算苦涩难懂，还是比较简单的哈，嗯，效果图看看就好，别做那么多右手螺旋定则哈....

3.代码示例下载：

DrawDemo1.zip 项目比较大，20多M，图片资源比较多，你懂的~

本节小结：

>  好的，本节写了关于绘图的两个小例子，还是蛮有趣的，相信你发下了，擦美女衣服那里， 消除的时候是方块的，不那么完美是吧，没事，下节我们学多个PorterDuff这个东西，我们再来写多个例子，相比起这个代码就简单很多了，另外，时间关系，代码并没有去优化或者整理，可以根据自己需求进行修改~好的，就说这么多，祝大家周末愉快~

8.3.3 Paint API之—— MaskFilter(面具)

本节引言：

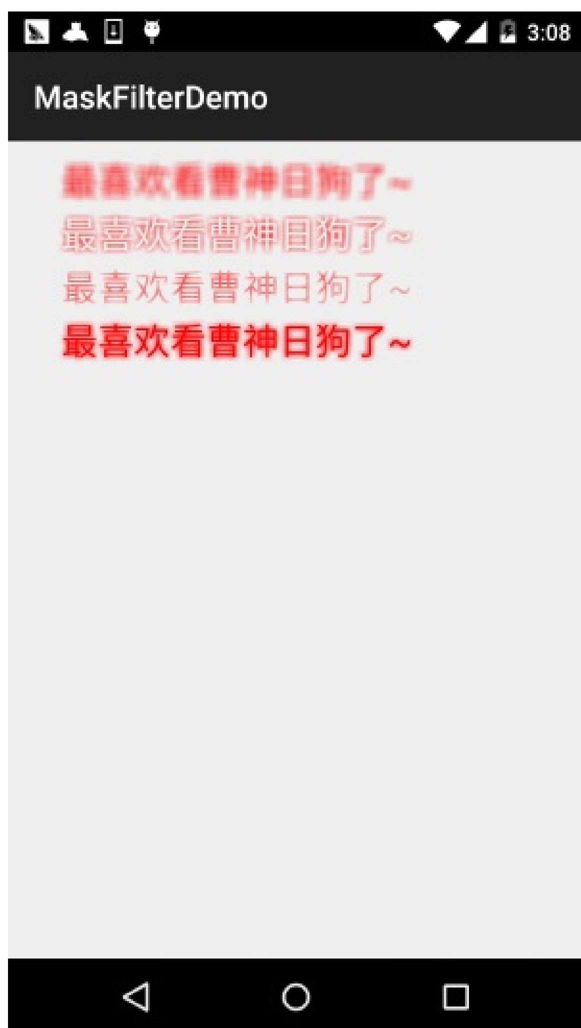
> 在Android基础入门教程——8.3.1 三个绘图工具类详解的Paint方法中有这样一个方法：
>
> **setMaskFilter(MaskFilter maskfilter)**： 设置MaskFilter，可以用不同的MaskFilter实现滤镜的效果，如滤化，立体等！ 而我们一般不会直接去用这个MaskFilter，而是使用它的两个子类：
>
> **BlurMaskFilter**：指定了一个模糊的样式和半径来处理Paint的边缘。
>
> **EmbossMaskFilter**：指定了光源的方向和环境光强度来添加浮雕效果。 下面我们来写个例子来试验一下~！
>
> 官方API文档： BlurMaskFilter； EmbossMaskFilter；

1.BlurMaskFilter(模糊效果)

> 说什么滤镜立体，谁知道怎么样，示例见真知：

代码示例：

运行效果图：



实现代码：

> 这里我们创建一个自定义View，在里面完成绘制！

BlurMaskFilterView.java：

```
/**
 * Created by Jay on 2015/10/21 0021.
 */
public class BlurMaskFilterView extends View{

    public BlurMaskFilterView(Context context) {
        super(context);
    }

    public BlurMaskFilterView(Context context, AttributeSet attrs)
        super(context, attrs);
    }

    public BlurMaskFilterView(Context context, AttributeSet attrs,
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onDraw(Canvas canvas) {

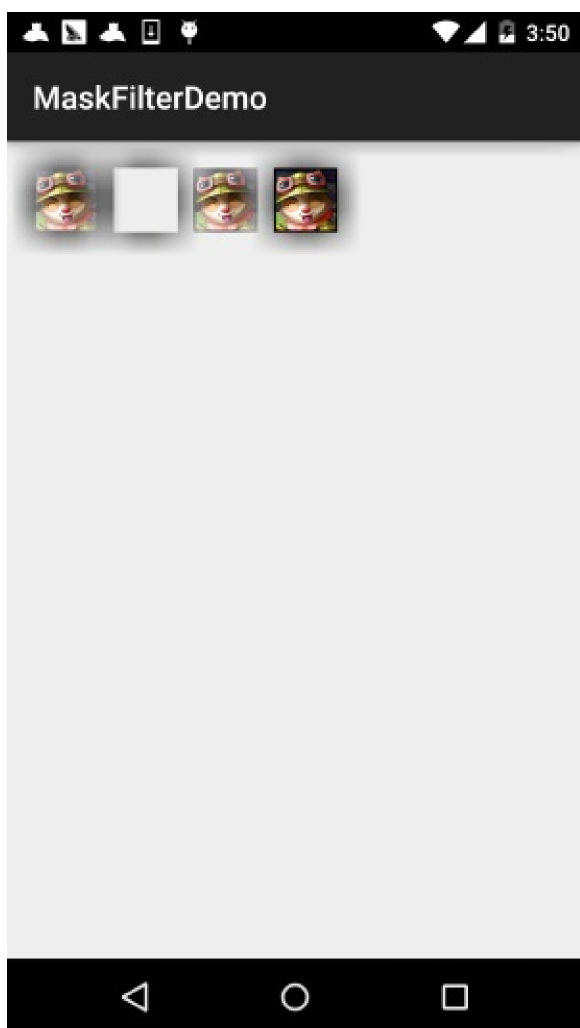
        BlurMaskFilter bmf = null;
        Paint paint=new Paint();
        paint.setAntiAlias(true);          //抗锯齿
        paint.setColor(Color.RED);//画笔颜色
        paint.setStyle(Paint.Style.FILL);  //画笔风格
        paint.setTextSize(68);             //绘制文字大小，单位px
        paint.setStrokeWidth(5);           //画笔粗细

        bmf = new BlurMaskFilter(10f,BlurMaskFilter.Blur.NORMAL);
        paint.setMaskFilter(bmf);
        canvas.drawText("最喜欢看曹神日狗了~", 100, 100, paint);
        bmf = new BlurMaskFilter(10f,BlurMaskFilter.Blur.OUTER);
        paint.setMaskFilter(bmf);
        canvas.drawText("最喜欢看曹神日狗了~", 100, 200, paint);
        bmf = new BlurMaskFilter(10f,BlurMaskFilter.Blur.INNER);
        paint.setMaskFilter(bmf);
        canvas.drawText("最喜欢看曹神日狗了~", 100, 300, paint);
        bmf = new BlurMaskFilter(10f,BlurMaskFilter.Blur.SOLID);
        paint.setMaskFilter(bmf);
        canvas.drawText("最喜欢看曹神日狗了~", 100, 400, paint);

        setLayerType(View.LAYER_TYPE_SOFTWARE, null);     //关闭硬件
    }
}
```

> 好的，从上面的代码示例，我们可以发现，我们使用这个BlurMaskFilter，无非是， 在构造方法中实例化：
>
> **BlurMaskFilter(10f,BlurMaskFilter.Blur.NORMAL);**
>
> 我们可以控制的就是这两个参数：
>
> 第一个参数：指定模糊边缘的半径；
>
> 第二个参数：指定模糊的风格，可选值有：
>
> - BlurMaskFilter.Blur.**NORMAL**：内外模糊
> - BlurMaskFilter.Blur.**OUTER**：外部模糊
> - BlurMaskFilter.Blur.**INNER**：内部模糊
> - BlurMaskFilter.Blur.**SOLID**：内部加粗，外部模糊

可能还是有点不清晰，我们找个图片来试试：



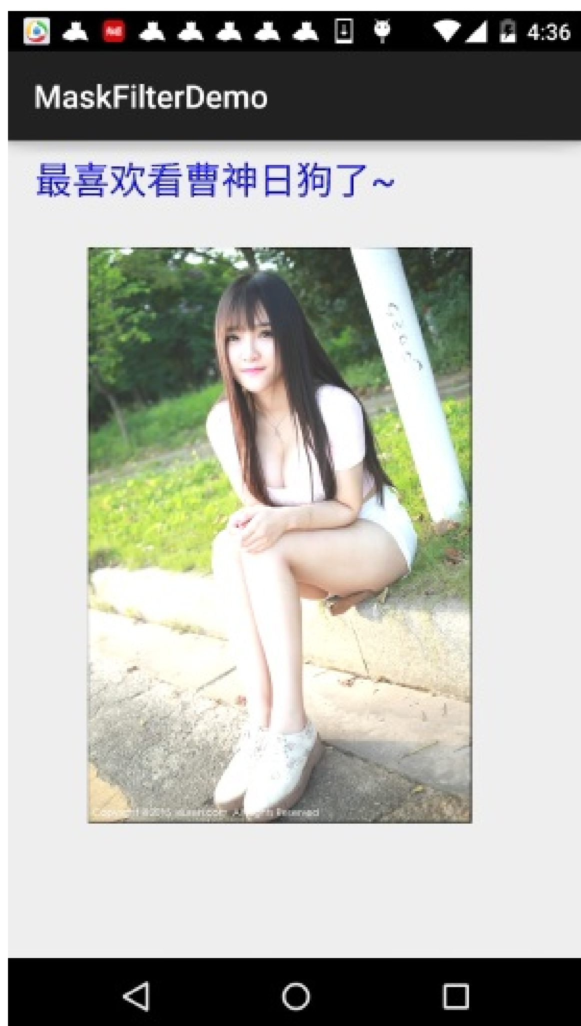
这里我们把模糊半径修改成了50，就更加明显了~

2.EmbossMaskFilter(浮雕效果)

> 如题，通过指定环境光源的方向和环境光强度来添加浮雕效果，同样，我们写个示例来看看效果：

代码示例：

运行效果图：



实现代码：

```
/**
 * Created by Jay on 2015/10/22 0022.
 */
public class EmbossMaskFilterView extends View{

    public EmbossMaskFilterView(Context context) {
        super(context);
    }

    public EmbossMaskFilterView(Context context, AttributeSet attrs
        super(context, attrs);
    }

    public EmbossMaskFilterView(Context context, AttributeSet attrs
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        float[] direction = new float[]{ 1, 1, 3 };   // 设置光源的方
        float light = 0.4f;     //设置环境光亮度
        float specular = 8;     // 定义镜面反射系数
        float blur = 3.0f;      //模糊半径
        EmbossMaskFilter emboss=new EmbossMaskFilter(direction,ligh

        Paint paint = new Paint();
        paint.setAntiAlias(true);          //抗锯齿
        paint.setColor(Color.BLUE);//画笔颜色
        paint.setStyle(Paint.Style.FILL);  //画笔风格
        paint.setTextSize(70);             //绘制文字大小，单位px
        paint.setStrokeWidth(8);           //画笔粗细
        paint.setMaskFilter(emboss);

        paint.setMaskFilter(emboss);
        canvas.drawText("最喜欢看曹神日狗了~", 50, 100, paint);

        Bitmap bitmap = BitmapFactory.decodeResource(getResources()
        canvas.drawBitmap(bitmap, 150, 200, paint);

        setLayerType(View.LAYER_TYPE_SOFTWARE, null);     //关闭硬件
    }
}
```

> 从效果图我们就可以看出一些EmbossMaskFilter的效果，修改光线，形成浮雕效果~妹子图不明显， 文字就很清晰显示出纹路了！和BlurMaskFilter一样，相关的设置都是在构造方法中进行！
>
> **EmbossMaskFilter(float[] direction, float ambient, float specular, float blurRadius)** 参数依次是：
>
> **direction**：浮点型数组，用于控制x,y,z轴的光源方向
>
> **ambient**：设置环境光亮度，0到1之间
>
> **specular**：镜面反射系数
>
> **blurRadius**：模糊半径
>
> 你可以修改这些值，试试不同的效果，比如我修改下上述的，又会是另一种效果：
>
> //这里为了明显点，换成了绿色
>
> 最喜欢看曹神日狗了~

3.注意事项

> 在使用MaskFilter的时候要注意，当我们的targetSdkVersion >= 14的时候，MaskFilter 就不会起效果了，这是因为Android在API 14以上版本都是默认开启硬件加速的，这样充分 利用GPU的特性，使得绘画更加平滑，但是会多消耗一些内存！好吧，我们把硬件加速关了 就好，可以在不同级别下打开或者关闭硬件加速，一般是关闭~
>
> - **Application**：在配置文件的application节点添加：android:hardwareAccelerated="true"
> - **Activity**：在配置文件的activity节点添加 android:hardwareAccelerated="false"
> - **View**：可以获得View对象后调用，或者直接在View的onDraw()方法里设置： view.setLayerType(View.LAYER_TYPE_HARDWARE, null);

示例代码下载：

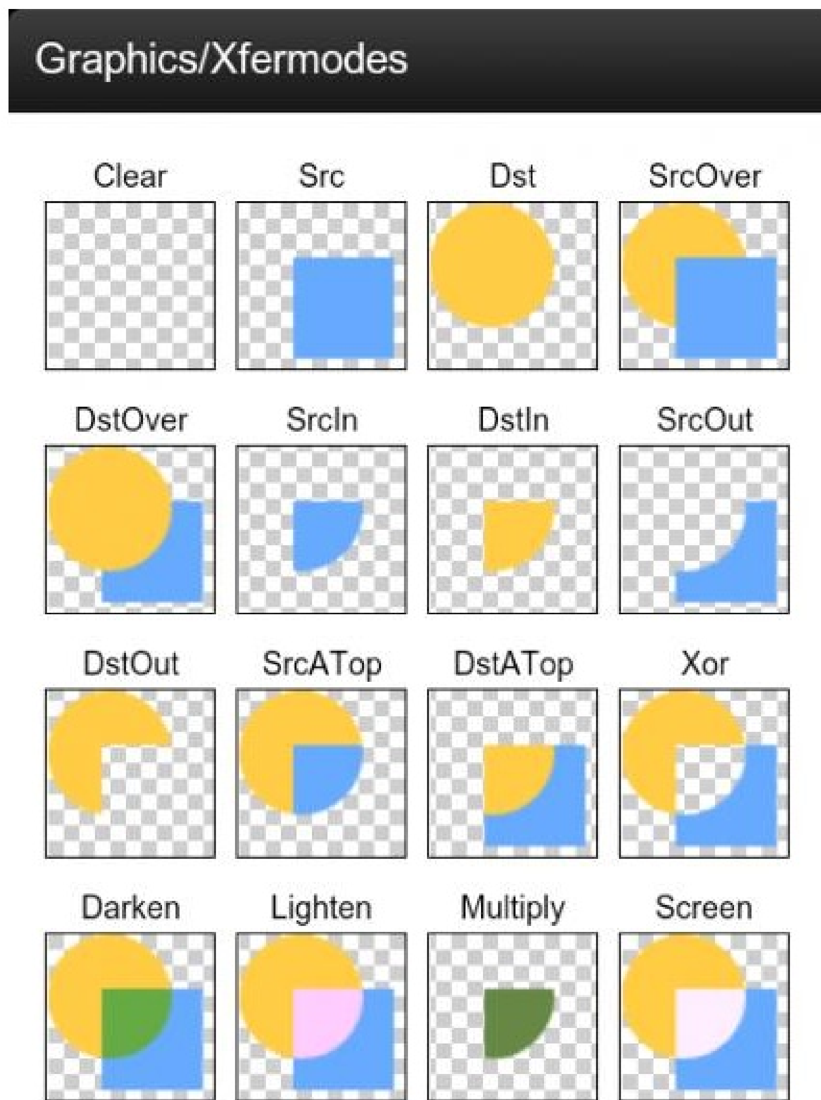
MaskFilterDemo.zip

本节小结：

> 本节给大家演示了Paint的一个API，**setMaskFilter(MaskFilter maskfilter)**，学习了 MaskFilter两个子类的基本用法：BlurMaskFilter(模糊效果)与EmbossMaskFilter(浮雕效果)， 比较简单，多学一点，对我们进阶部分的自定义控件也是没有坏处的~好的，就说这么多， 谢谢~
>
> 对了，忘了说，其实在SDK中的example中有个类，就演示了这两种用法：
>
> **samples\android-xx\legacy\ApiDemos\src\com\example\android\apis\graphics** 目录下的：**FingerPaint.java**文件~

8.3.4 Paint API之—— Xfermode与PorterDuff详解(一)

本节引言：

> 不知道标题这两个玩意你熟不熟悉啦，如果自己实现过圆角或者圆形图片，相信对这两个名词 并不模式，一时半伙没想起来？没关系，下面这个图你可曾见过？
>
> 
>
>
> PS：网上都说在：\samples\android-XX\legacy\ApiDemos\src\com\example\android\apis\graphics 下能找到这个图片- -，然而并没有，不知道是不是因为我的sample是android-22的，只在这里找到一个 **Xfermodes.java**的Java文件！这里直接贴下网上找到的~

> 嗯，说回来，这图相信大部分朋友都见过吧，没见过也没关系，本节我们带大家来一点点的学习 这个东西~，看回我们前面的Android基础入门教程——8.3.1 三个绘图工具类详解
>
> **setXfermode**(Xfermode xfermode)：设置图形重叠时的处理方式，如合并，取交集或并集， 经常用来制作橡皮的擦除效果！
>
> 我们来到官方文档：Xfermode，我们发现他有三个儿子：
>
> public class
>
> Xfermode
>
> extends Object
>
> java.lang.Object
> ↳android.graphics.Xfermode
>
> ▸ Known Direct Subclasses
>
> AvoidXfermode, PixelXorXfermode, PorterDuffXfermode
>
> 本节我们来学习他的前两个儿子~

大儿子：AvoidXfermode

> 嗯，和前面学的MaskFilter的两个子类一样，不支持硬件加速，所以如果是API 14以上的版本， 需要关闭硬件加速才会有效果！怎么关自己看上一节哈~
>
> 我们来看看他给我们提供的构造方法！官方API文档：AvoidXfermode
>
> | Public Constructors |
> | --- |
> | AvoidXfermode(int opColor, int tolerance, AvoidXfermode.Mode mode)
This xfermode draws, or doesn't draw, based on the destination's distance from an op-color. |
>
> 参数有三个，依次是：
>
> **opColor**：一个十六进制的带透明度的颜色值，比如0x00C4C4;
>
> **tolerance**：容差值，如果你学过PS可能用过魔棒工具，就是设置选取颜色值的范围，比如 容差为0，你选的是纯黑的小点，当容差调为40的时候，范围已经扩大到大块黑色这样！如果 还不是很明白，等下我们写写代码就知道了！
>
> **mode**：AvoidXfermode模式，有两种：**TARGET**与**AVOID**

模式1：AvoidXfermode.Mode.TARGET

> 该模式会判断画布上是否有与我们设置颜色值不一样的颜色，如果有的话，会把这些区域 染上一层画笔定义的颜色，其他地方不染色！下面我们写代码演示下，顺便让大家感觉下 这个容差值！

使用代码示例：

运行效果图：

嗯，先上下原图，素材来自**gank.io**：



接下来我们随便把墙上某个地方的颜色用颜色取色器取下，然后写一个简单的View!

PS:需要在AndroidManifest.xml中的appliction节点添加关闭硬件加速：**android:hardwareAccelerated="false"**

```
/**
 * Created by Jay on 2015/10/22 0022.
 */
public class AvoidXfermodeView1 extends View {

    private Paint mPaint;
    private Bitmap mBitmap;
    private AvoidXfermode avoidXfermode;

    public AvoidXfermodeView1(Context context) {
        super(context);
        init();
    }

    public AvoidXfermodeView1(Context context, AttributeSet attrs)
        super(context, attrs);
        init();
    }

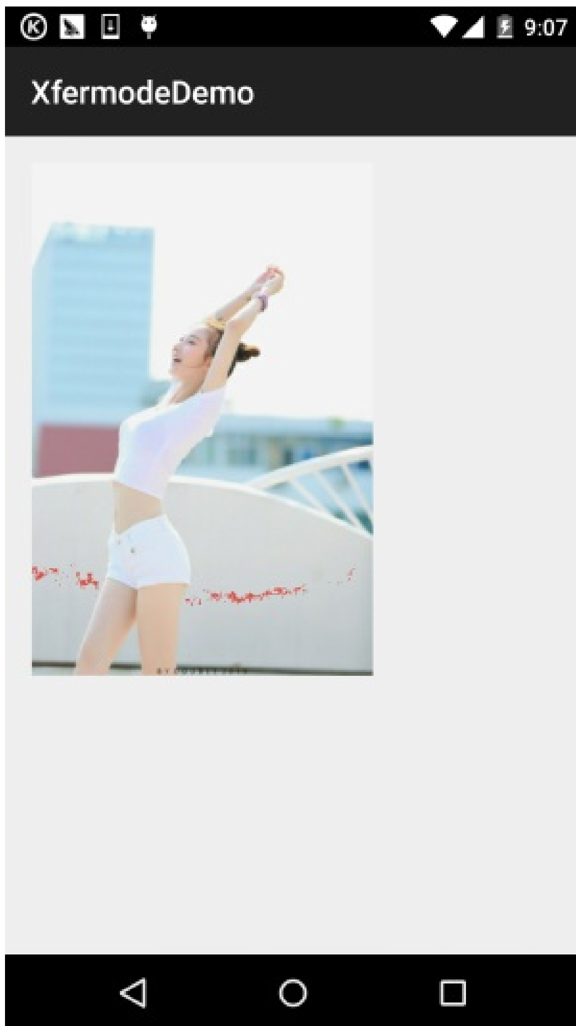
    public AvoidXfermodeView1(Context context, AttributeSet attrs,
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  //抗锯齿
        avoidXfermode = new AvoidXfermode(0XFFCCD1D4, 0, AvoidXferm
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi

    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(mBitmap, 50, 50, mPaint);
        mPaint.setARGB(255, 222, 83, 71);
        mPaint.setXfermode(avoidXfermode);
        canvas.drawRect(50, 50, 690, 1010, mPaint);
    }
}
```

运行后的效果：

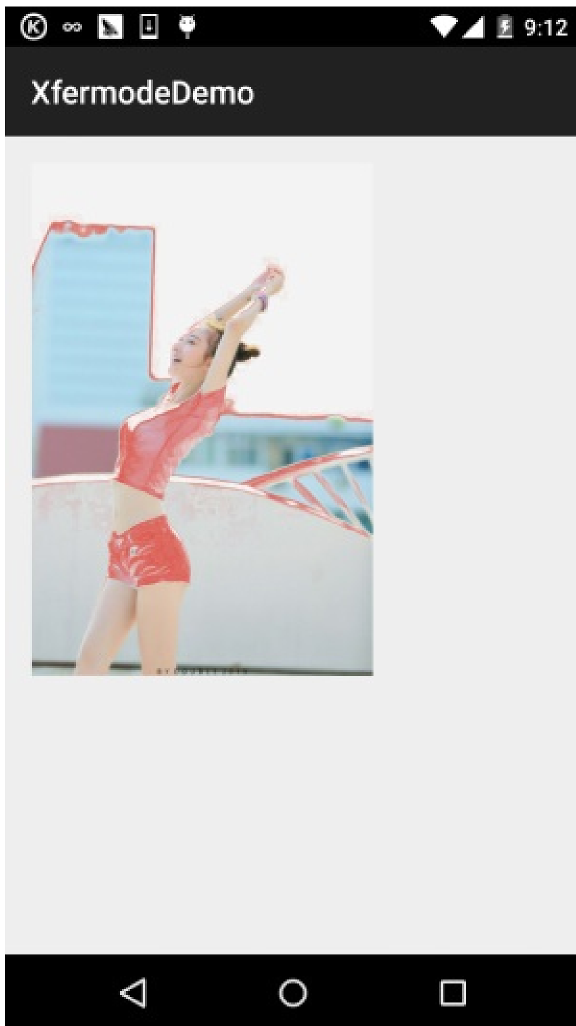


看到墙上那堆姨妈红了没，效果杠杠的，这里我们的容差值并没有发挥作用，我们改一改，把 妹子的白衣服变成姨妈红！

我们把上面构造AvoidXfermode的内容改成：

```
avoidXfermode = new AvoidXfermode(0XFFD9E5F3, 25, AvoidXfermode.Moc
```

然后，妹子身上的白衣服就变成姨妈红了... ，满满的罪恶感...

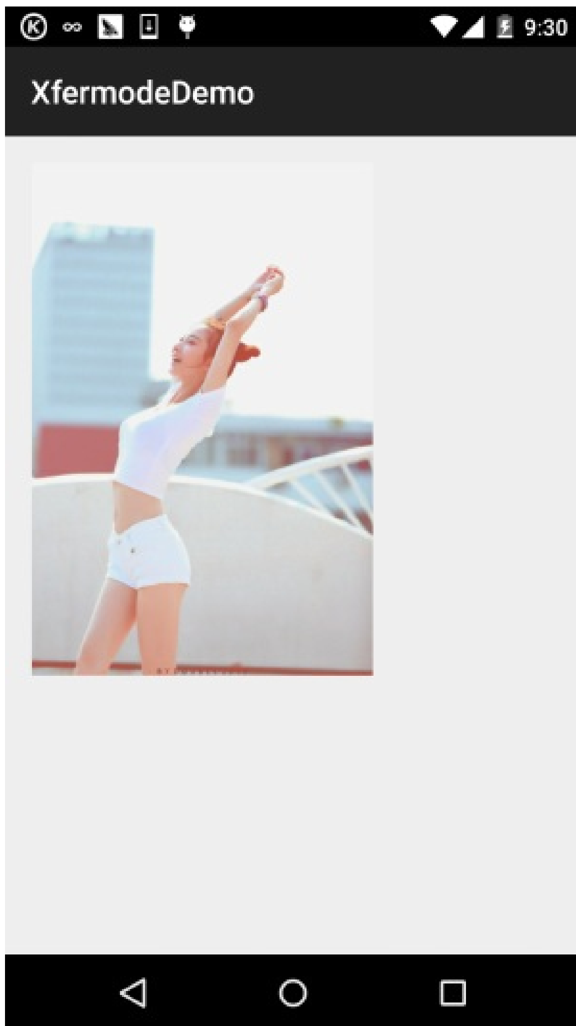


模式2：AvoidXfermode.Mode.AVOID

> 和上面的TARGET模式相反，上面是颜色一样才改变颜色，这里是颜色不一样反而改变颜色， 而容差值同样带来相反的结果，容差值为0时，只有当图片中的像素颜色值与设置的颜色值完全不一样 的时候才会被染色，而当容差值达到最大值255的时候，稍微有一点颜色不一样就会被染色！ 我们只需简单的修改上面的例子就可以了， 同一是修改下构造AvoidXfermode的内容！ 我们改成下面这句：

```
avoidXfermode = new AvoidXfermode(0XFFD9E5F3,230, AvoidXfermode.Moc
```

运行效果图：

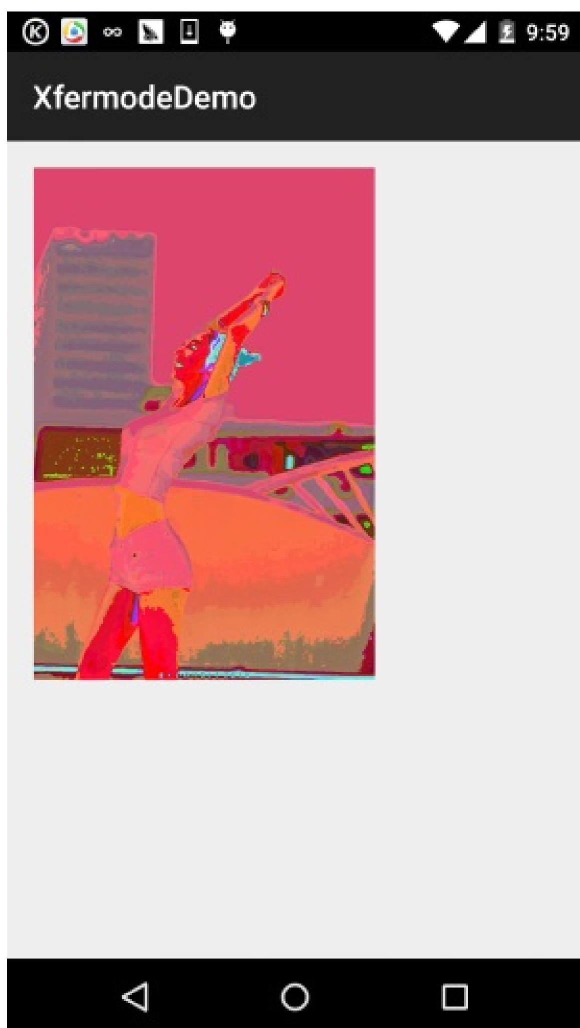


二儿子：PixelXorXfermode

> 这个则是另一种图像混排模式，比起大儿子更简单，他的构造方法如下：
>
> 官方API文档：PixelXorXfermode
>
> | Public Constructors | |
> |---|---|
> | | PixelXorXfermode(int opColor) |
>
> 参数解析：
>
> 就一个16进制带透明值得颜色值，至于这个值的作用，是有一个算法的：PixelXorXfermode内部是按照" opColor ^ src ^ dst "这个异或算法运算的，得到一个不透明的(alpha = 255)的色彩值，设置到图像中！好吧，这是网上搜的具体我也不知道，写个例子试试效果呗~

代码示例：

运行效果图：



实现代码：

```
/**
 * Created by Jay on 2015/10/22 0022.
 */
public class PixelXorXfermodeView1 extends View{

    private Paint mPaint;
    private Bitmap mBitmap;
    private PixelXorXfermode pixelxorXfermode;

    public PixelXorXfermodeView1(Context context) {
        super(context);
        init();

    }

    public PixelXorXfermodeView1(Context context, AttributeSet attı
        super(context, attrs);
        init();
    }

    public PixelXorXfermodeView1(Context context, AttributeSet attı
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);  //抗锯齿
        pixelxorXfermode = new PixelXorXfermode(0XFFD9E5F3);
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(mBitmap, 50, 50, mPaint);
        mPaint.setARGB(255, 222, 83, 71);
        mPaint.setXfermode(pixelxorXfermode);
        canvas.drawRect(50, 50, 690, 1010, mPaint);
    }

}
```

本节示例代码下载：

XfermodeDemo.zip

本节小结：

> 好吧，满满的罪恶感，很漂亮的一个妹子，结果给我写demo写成了这个样子，别怪我，嗯，对了，忘记说，Xfermode的大儿子和二儿子已经过世(过期)，在API 16后的版本，就 过期了，也就说本节并没什么卵用...
>
> 也不能这样说，Apache在4.4后的版本都给阉割了，但是还是有人用着HttpClient，或者 由这个库写的HTTP请求框架哈~当然，这种人基本很少很少！不过学多点总没坏处，是吧， 下节的三儿子**PorterDuffXfermode**就没过时啦，也很重要，嗯，放心，不会又会毁照片！ 嗯，就说这么多，谢谢~

8.3.5 Paint API之—— Xfermode与PorterDuff详解(二)

本节引言：

> 上一节，我们学习了Xfermode两个已经过世(过时)的儿子：**AvoidXfermode, PixelXorXfermode**，虽然说有点用，但是终归是被淘汰的了，本节我们来学习Xfermode还健在的三儿子：**PorterDuffXfermode**；
>
> 先祭上官方API文档：PorterDuffXfermode！文档内容很少，我们可以看到他的构造方法：
>
> | Public Constructors | |
> |---|---|
> | | PorterDuffXfermode(PorterDuff.Mode mode)
Create an xfermode that uses the specified porter-duff mode. |
>
> 参数只有一个：**PorterDuff.Mode** mode，而Android给我们提供了16种图片混排模式，简单点可以 理解为两个图层按照不同模式，可以组合成不同的结果显示出来！16种混排模式的结果图如下：

> 
>
>
> 这里两个图层：先绘制的图是目标图**(DST)**，后绘制的图是源图**(SRC)**！
>
> 当然，在文档中我们发现可供使用的模式并不是16种，而是18种，新增了**ADD**和**OVERLAY**两种模式!
>
> 嗯，说多也白说，代码最实际，本节我们写下代码来验证下这18种模式吧！
> 
>
> PS:这个PorterDuff的命名其实是两个人名的组合：Tomas Proter和 Tom Duff组成的，他们是最早在 最早在SIGGRAPH上提出图形混合概念的大神级人物，有兴趣的自行百度~

写个例子来验证上面的这个图：

> 好的，我们来写个例子验证下上面这个图，通过修改不同的模式，来对结果进行对比分析！

代码实现：

Step 1：我们先写个获取屏幕宽高的工具类吧！**ScreenUtil.java**：

```
/**
 * Created by Jay on 2015/10/23 0023.
 */
public class ScreenUtil {
    /**
     * 获取屏幕宽高，sdk17后不建议采用
     *
     * @param context
     */
    public static int[] getScreenHW(Context context) {
        WindowManager manager = (WindowManager) context.getSystemSe
        Display display = manager.getDefaultDisplay();
        int width = display.getWidth();
        int height = display.getHeight();
        int[] HW = new int[] { width, height };
        return HW;
    }

    /**
     * 获取屏幕宽高，建议采用
     *
     * @param context
     */
    public static int[] getScreenHW2(Context context) {
        WindowManager manager = (WindowManager) context.getSystemSe
        DisplayMetrics dm = new DisplayMetrics();
        manager.getDefaultDisplay().getMetrics(dm);
        int width = dm.widthPixels;
        int height = dm.heightPixels;
        int[] HW = new int[] { width, height };
        return HW;
    }

    /**
     * 获取屏幕的宽度
     *
     * @param context
     * @return
     */
    public static int getScreenW(Context context) {
        return getScreenHW2(context)[0];
    }

    /**
     * 获取屏幕的高度
     *
     * @param context
     * @return
     */
    public static int getScreenH(Context context) {
```

```
        return getScreenHW2(context)[1];
    }
}
```

Step 2：编写我们的自定义View类，在这里做试验！**XfermodeView.java**：

```
/**
 * Created by Jay on 2015/10/23 0023.
 */
public class XfermodeView extends View {

    private PorterDuffXfermode pdXfermode;   //定义PorterDuffXfermo
    //定义MODE常量，等下直接改这里即可进行测试
    private static PorterDuff.Mode PD_MODE = PorterDuff.Mode.ADD;
    private int screenW, screenH; //屏幕宽高
    private int width = 200;      //绘制的图片宽高
    private int height = 200;
    private Bitmap srcBitmap, dstBitmap;     //上层SRC的Bitmap和下层D

    public XfermodeView(Context context) {
        this(context, null);
    }

    public XfermodeView(Context context, AttributeSet attrs) {
        super(context, attrs);
        screenW = ScreenUtil.getScreenW(context);
        screenH = ScreenUtil.getScreenH(context);
        //创建一个PorterDuffXfermode对象
        pdXfermode = new PorterDuffXfermode(PD_MODE);
        //实例化两个Bitmap
        srcBitmap = makeSrc(width, height);
        dstBitmap = makeDst(width, height);
    }

    public XfermodeView(Context context, AttributeSet attrs, int de
        super(context, attrs, defStyleAttr);
    }

    //定义一个绘制圆形Bitmap的方法
    private Bitmap makeDst(int w, int h) {
        Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_88
        Canvas c = new Canvas(bm);
        Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);
        p.setColor(0xFF26AAD1);
        c.drawOval(new RectF(0, 0, w * 3 / 4, h * 3 / 4), p);
        return bm;
    }

    //定义一个绘制矩形的Bitmap的方法
    private Bitmap makeSrc(int w, int h) {
```

```
        Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_88
        Canvas c = new Canvas(bm);
        Paint p = new Paint(Paint.ANTI_ALIAS_FLAG);
        p.setColor(0xFFFFCE43);
        c.drawRect(w / 3, h / 3, w * 19 / 20, h * 19 / 20, p);
        return bm;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setFilterBitmap(false);
        paint.setStyle(Paint.Style.FILL);
        canvas.drawBitmap(srcBitmap, (screenW / 3 - width) / 2, (sc
        canvas.drawBitmap(dstBitmap, (screenW / 3 - width) / 2 + sc

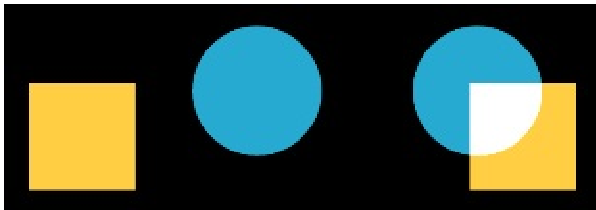
        //创建一个图层，在图层上演示图形混合后的效果
        int sc = canvas.saveLayer(0, 0, screenW, screenH, null, Can
                Canvas.CLIP_SAVE_FLAG |
                Canvas.HAS_ALPHA_LAYER_SAVE_FLAG |
                Canvas.FULL_COLOR_LAYER_SAVE_FLAG |
                Canvas.CLIP_TO_LAYER_SAVE_FLAG);

        canvas.drawBitmap(dstBitmap, (screenW / 3 - width) / 2 + sc
                (screenH / 2 - height) / 2, paint);     //绘制i
        //设置Paint的Xfermode
        paint.setXfermode(pdXfermode);
        canvas.drawBitmap(srcBitmap, (screenW / 3 - width) / 2 + sc
                (screenH / 2 - height) / 2, paint);
        paint.setXfermode(null);
        // 还原画布
        canvas.restoreToCount(sc);
    }
}
```

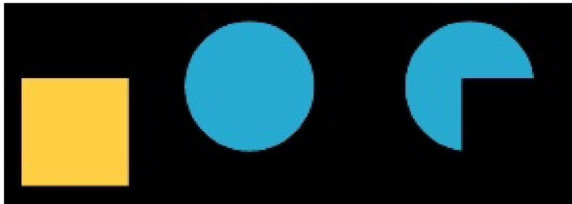
代码看起来好复杂是吧，其实不然，无非就是获取了屏幕宽高，然后画了一个矩形一个圆形， 计算了一下他们的位置，然后设置下图层(固定写法)，接着设下下画笔setXfermode，接着 绘制到canvas上而已，你看不懂的可能是绘制位置的计算吧，其实不然，位置你喜欢怎么定 都可以！那么接下来我们来一个个看下解果咯，你只需修改**PD_MODE**的值设置为不同模式即可！

运行效果图：

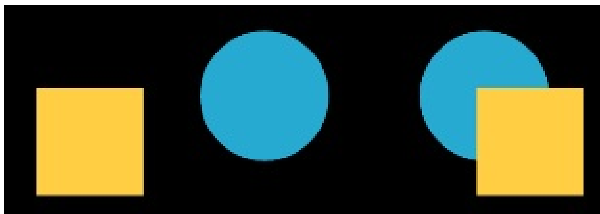
1）PorterDuff.Mode.ADD：

饱和度叠加

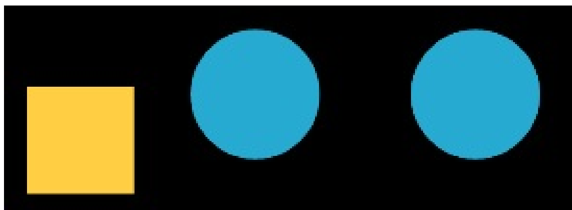
2）PorterDuff.Mode.CLEAR：

所绘制不会提交到画布上，嗯结果...不知道是为什么了，正常是没东西的..

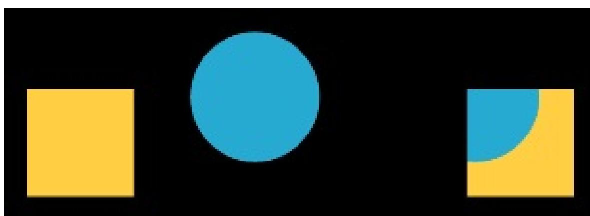
3）PorterDuff.Mode.DARKEN：

取两图层全部区域，交集部分颜色加深

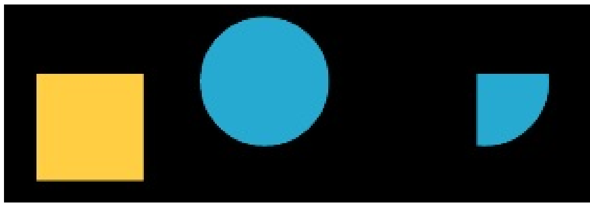
4）PorterDuff.Mode.DST：

只保留目标图的alpha和color，所以绘制出来只有目标图

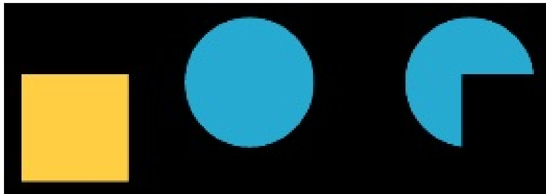
5）PorterDuff.Mode.DST_ATOP：

源图和目标图相交处绘制目标图，不相交的地方绘制源图

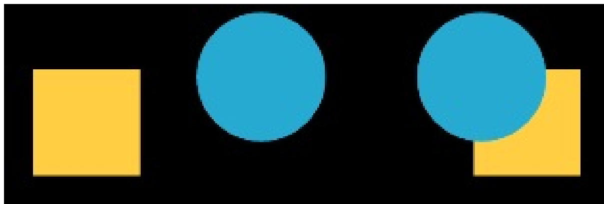
6）PorterDuff.Mode.DST_IN：


两者相交的地方绘制目标图，绘制的效果会受到原图处的透明度影响

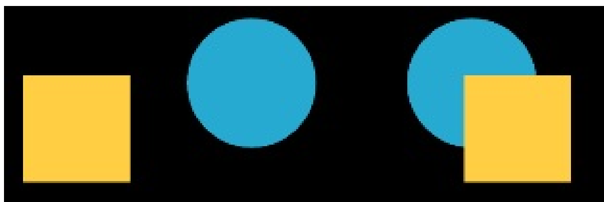
7）PorterDuff.Mode.DST_OUT：


在不相交的地方绘制目标图

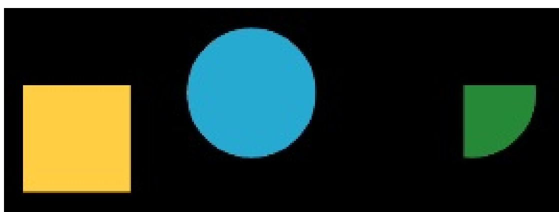
8）PorterDuff.Mode.DST_OVER：


目标图绘制在上方

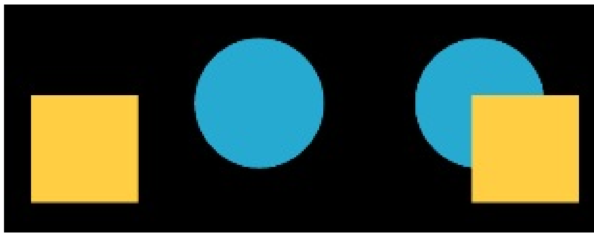
9）PorterDuff.Mode.LIGHTEN：


取两图层全部区域，点亮交集部分颜色

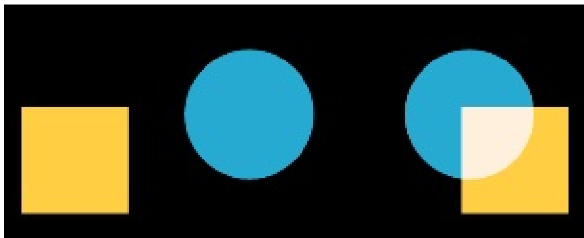
10）PorterDuff.Mode.MULTIPLY：


取两图层交集部分叠加后颜色

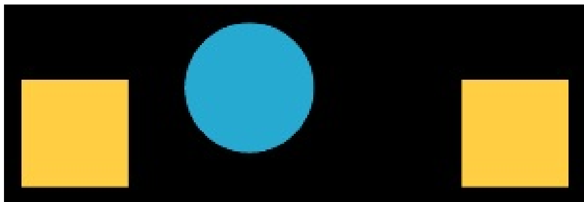
11）PorterDuff.Mode.OVERLAY：


叠加

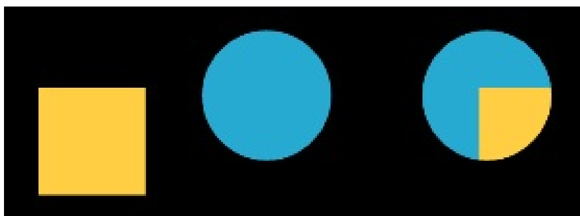
12）PorterDuff.Mode.SCREEN：


取两图层全部区域，交集部分变为透明色

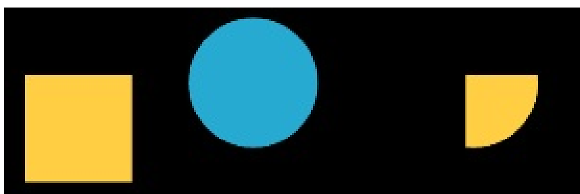
13）PorterDuff.Mode.SRC：


只保留源图像的alpha和color，所以绘制出来只有源图

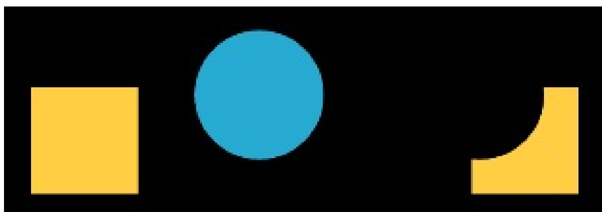
14）PorterDuff.Mode.SRC_ATOP：


源图和目标图相交处绘制源图，不相交的地方绘制目标图

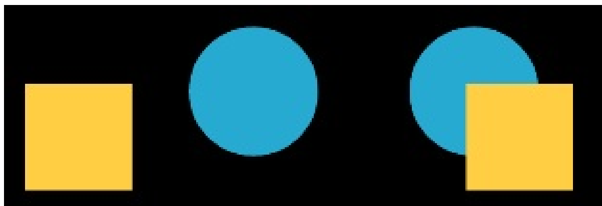
15）PorterDuff.Mode.SRC_IN：


两者相交的地方绘制源图

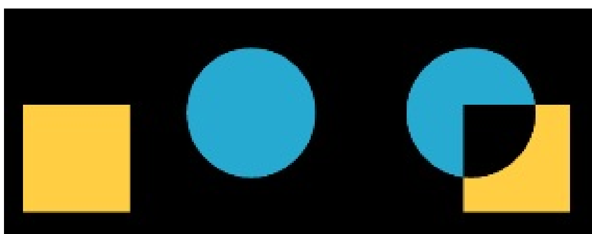
16）PorterDuff.Mode.SRC_OUT：


不相交的地方绘制源图

17）PorterDuff.Mode.SRC_OVER：


把源图绘制在上方

18）PorterDuff.Mode.XOR：


不相交的地方按原样绘制源图和目标图

本节示例代码下载：

PorterDuffXfermodeDemo.zip

本节小结：

> 嗯，本节就写了一个简单的View来验证这18种不同PorterDuff.Mode下的不同效果，嘿嘿，蛮耗时间的，不过，读者看起来肯定清晰多了是吧~当然，这只是一些初步的见解！
>
> **PorterDuffXfermode的PorterDuff.Mode**对于我们自定义控件是非常重要的！本节我们初步了解，下节我们挑几个例子来练练手！
>
> 如果你想看关于PorterDuff.Mode更加详细的介绍可见：Android Paint之setXfermode PorterDuffXfermode 讲解，别人写的不错的一篇文章！嗯，就到这里，明早体检，今天就写这么多~ 

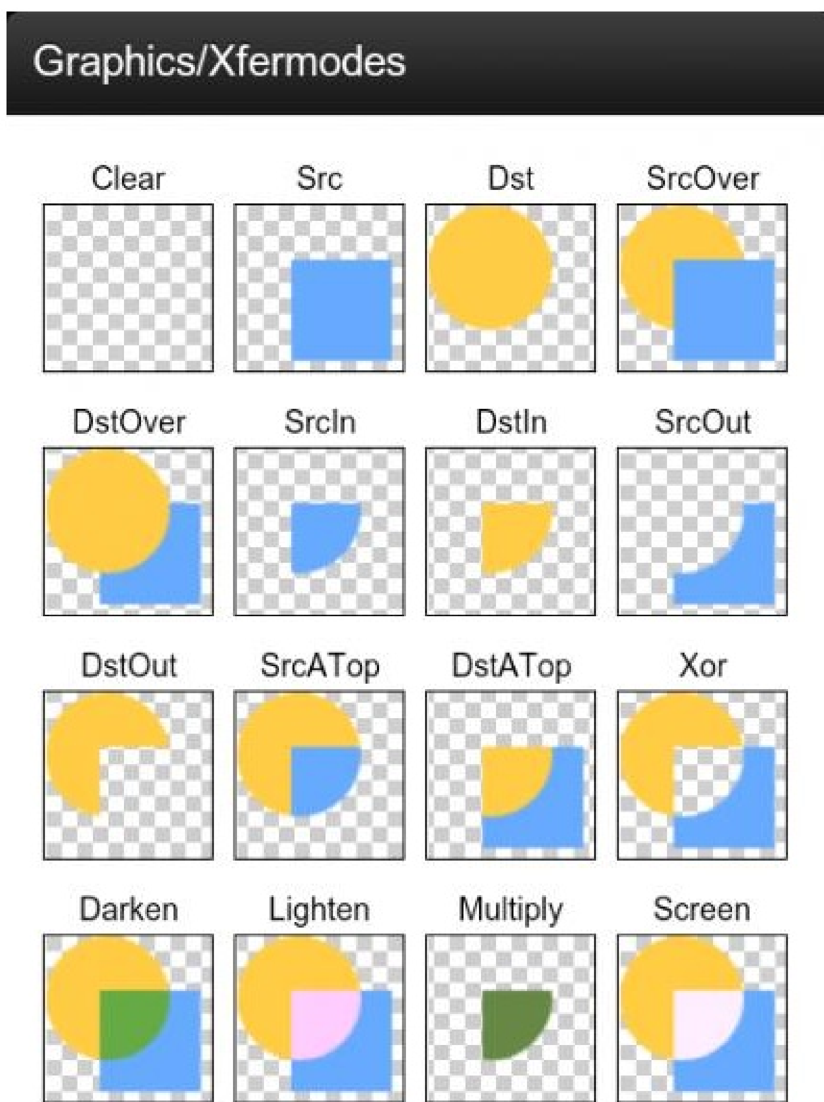
8.3.6 Paint API之—— Xfermode与PorterDuff详解(三)

本节引言：

上一节，我们学习了Xfermode中的三儿子：PorterDuffXfermode构造方法中的为一个参数：**PorterDuff.Mode**，我们在观看了16种图片混排模式后，又自己写代码来验证了一下文档中 18种不同的混排模式，18种是新增了ADD和OVERLAY两种模式！当然，仅仅验证知道是不够的，本节我们来写个例子，帮助我们熟悉下实际当中我们如何去使用PorterDuff.Mode为我们提供的 这些混排模式！本节带来的例子是：圆形&圆角图形的实现！

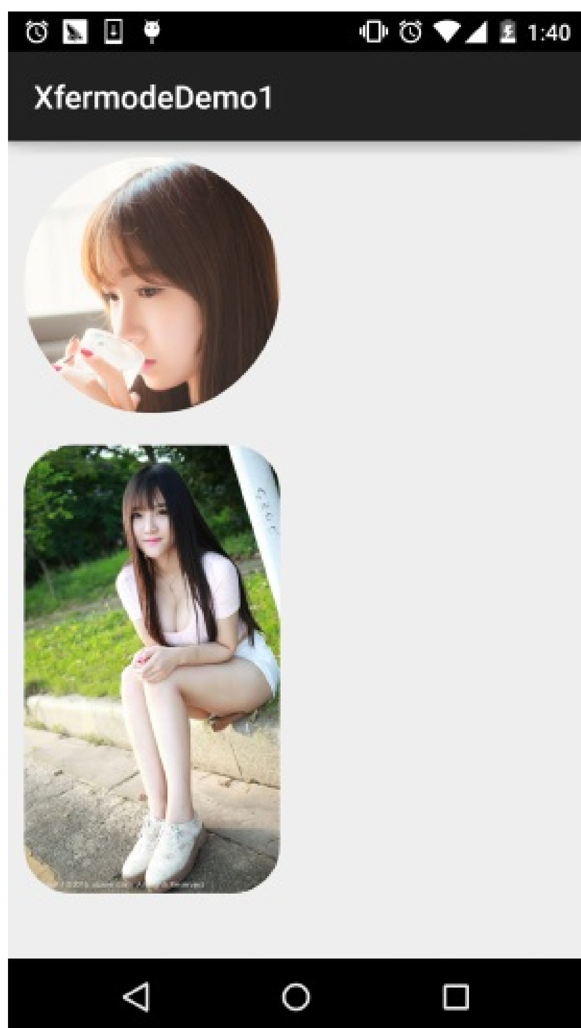
在2.3.4 ImageView(图像视图)我们最后就讲解了一个最简单 绘制圆形ImageView的实现，原理是在图片上调用clipPath切出一个圆形！

而这节则是利用**PorterDuff.Mode**中的**DST_IN**模式来实现,话不多说，开始本节内容！ PS：本节例子采自弘洋大神的——Android Xfermode 实战 实现圆形、圆角图片 另外，还是要贴下PorterDuff.Mode的效果图：



1.要实现的效果图以及实现流程分析：

运行后的效果图：



嗯，上述就是我们要实现的一个效果，通过这个**PorterDuff.Mode.DST_IN**模式来实现！我们来分析分析实现流程：

> - **Step 1**： Xfermode无非是两层图构成，先绘制的叫DST图(目标图)，后绘制的叫SRC图(原图)，我们要实现 圆形或者圆角，我们可以先把要显示的图片绘制出来(DST)，这里我们通过src的属性进行了设置；接着再绘制出圆形和圆角(SRC)，我们想显示的部分是他们相交的地方，而且是图片部分的内容， 所以选择：**DST_IN**模式！
> - **Step 2**： 嗯，知道了原理，接下来我们要考虑自定义ImageView相关的问题了：
> - 我们是想绘制的View是圆角或者圆形，那就需要加个属性来判断，而圆角也需要一个圆角半径的 参数，于是乎我们可以通过自定义属性(attrs.xml)的方式，然后再自定义View的构造方法中，将 这些参数取出来！
> - 接着到图片大小的计算了： 首先假如我们设置的是圆形的话，则需要让宽高一致，以最小值为准，我们可以在onMesure()方法 调用getMeasuredXxx()获得宽高，看谁小一点，调用setMeasuredDimension(x, x)；设置宽高！ 然后，我们在onDraw()方法中获取图片宽高，接着按照图片宽高，以及View宽高，计算缩放比例， 假如图片宽高与View的宽高不匹配，所犯后的图片宽高一定要大于View的宽高，so，取大值！
> - 再接着就到图片的绘制了，定义一个绘制图形的方法，接着初始化画笔后，设置setXfermode为 PorterDuff.Mode.DST_IN，先绘制图片，再绘制图形
> - 最后是图片缓存的一些东西，这里用了WeakReference来缓存图片，避免每次onDraw都分配内存 与重绘，最后在invalidate中清楚缓存！

大体的实现流程如上述，知道流程再看代码就简单很多了！

2.代码实现：

自定义控件属性：**res/attrs.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CircleImageView">
        <attr name="Radius" format="dimension"/>
        <attr name="type">
            <enum name="circle" value="0"/>
            <enum name="round" value="1"/>
        </attr>
    </declare-styleable>
</resources>
```

接着是自定义ImageView：**CircleImageView.java**：

```
/**
 * Created by Jay on 2015/10/25 0025.
 */
public class CircleImageView extends ImageView {
```

```
    private Paint mPaint;
    private Xfermode mXfermode = new PorterDuffXfermode(PorterDuff.
    private Bitmap mMaskBitmap;
    private WeakReference<Bitmap> mWeakBitmap;

    //图片相关的属性
    private int type;                           //类型，圆形或者圆角
    public static final int TYPE_CIRCLE = 0;
    public static final int TYPE_ROUND = 1;
    private static final int BODER_RADIUS_DEFAULT = 10;     //圆角默
    private int mBorderRadius;                  //圆角大小

    public CircleImageView(Context context) {
        this(context, null);
    }

    public CircleImageView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mPaint = new Paint();
        mPaint.setAntiAlias(true);
        //取出attrs中我们为View设置的相关值
        TypedArray tArray = context.obtainStyledAttributes(attrs, R
        mBorderRadius = tArray.getDimensionPixelSize(R.styleable.Ci
        type = tArray.getInt(R.styleable.CircleImageView_type, TYPE
        tArray.recycle();
    }

    public CircleImageView(Context context, AttributeSet attrs, int
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasur
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        if (type == TYPE_CIRCLE) {
            int width = Math.min(getMeasuredWidth(), getMeasuredHei
            setMeasuredDimension(width, width);    //设置当前View的大
        }
    }

    @Override
    protected void onDraw(Canvas canvas) {

        //在缓存中取出bitmap
        Bitmap bitmap = mWeakBitmap == null ? null : mWeakBitmap.ge
        if (bitmap == null || bitmap.isRecycled()) {
            //获取图片宽高
            Drawable drawable = getDrawable();
            int width = drawable.getIntrinsicWidth();
            int height = drawable.getIntrinsicHeight();

            if (drawable != null) {
                bitmap = Bitmap.createBitmap(getWidth(), getHeight(
```

```
                Canvas drawCanvas = new Canvas(bitmap);
                float scale = 1.0f;
                if (type == TYPE_ROUND) {
                    scale = Math.max(getWidth() * 1.0f / width, get
                            * 1.0f / height);
                } else {
                    scale = getWidth() * 1.0F / Math.min(width, hei
                }
                //根据缩放比例，设置bounds，相当于缩放图片了
                drawable.setBounds(0, 0, (int) (scale * width),
                        (int) (scale * height));

                drawable.draw(drawCanvas);
                if (mMaskBitmap == null || mMaskBitmap.isRecycled()
                    mMaskBitmap = getBitmap();
                }

                mPaint.reset();
                mPaint.setFilterBitmap(false);
                mPaint.setXfermode(mXfermode);

                //绘制形状
                drawCanvas.drawBitmap(mMaskBitmap, 0, 0, mPaint);

                //bitmap缓存起来，避免每次调用onDraw，分配内存
                mWeakBitmap = new WeakReference<Bitmap>(bitmap);

                //绘制图片
                canvas.drawBitmap(bitmap, 0, 0, null);
                mPaint.setXfermode(null);

            }
        }
        if (bitmap != null) {
            mPaint.setXfermode(null);
            canvas.drawBitmap(bitmap, 0.0f, 0.0f, mPaint);
            return;
        }
    }

    //缓存Bitmap，避免每次OnDraw都重新分配内存与绘图
    @Override
    public void invalidate() {
        mWeakBitmap = null;
        if (mWeakBitmap != null) {
            mMaskBitmap.recycle();
            mMaskBitmap = null;
        }
        super.invalidate();
    }

    //定义一个绘制形状的方法
```

```
    private Bitmap getBitmap() {
        Bitmap bitmap = Bitmap.createBitmap(getWidth(), getHeight()
                Bitmap.Config.ARGB_8888);

        Canvas canvas = new Canvas(bitmap);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);   //抗锯齿
        paint.setColor(Color.BLACK);
        if (type == TYPE_ROUND) {
            canvas.drawRoundRect(new RectF(0, 0, getWidth(), getHe
                    mBorderRadius, mBorderRadius, paint);
        } else {
            canvas.drawCircle(getWidth() / 2, getWidth() / 2, getW
        }
        return bitmap;
    }
}
```

最后在布局文件那里调用下:**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.jay.xfermodedemo1.CircleImageView
        android:layout_width="160dp"
        android:layout_height="240dp"
        android:layout_margin="10dp"
        android:src="@mipmap/ic_bg_meizi2"
        app:type="circle" />

    <com.jay.xfermodedemo1.CircleImageView
        android:layout_width="160dp"
        android:layout_height="280dp"
        android:layout_margin="10dp"
        android:src="@mipmap/ic_bg_meizi1"
        app:Radius="30dp"
        app:type="round" />

</LinearLayout>
```

好的，代码一次看不懂，看多两次就懂的了~

3.本节代码示例下载：

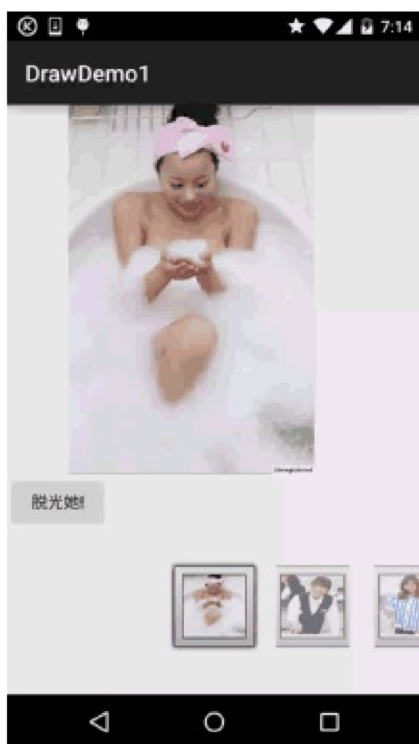
XfermodeDemo1.zip

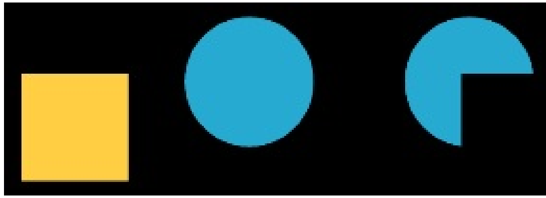
本节小结：

> 本节我们讲解了Xfermode与PorterDuff的第一个应用例子，设置DST_IN模式来实现 圆形和圆角图片ImageView的定制，相信大家对PorterDuff的简单应用已经有些眉目了， 打铁趁热，下一节我们同一会写个例子练练手~好的，就说这么多，谢谢~

8.3.7 Paint API之—— Xfermode与PorterDuff详解(四)

本节引言：

> 上节我们写了关于Xfermode与PorterDuff使用的第一个例子：圆角&圆形图片ImageView的实现， 我们体会到了PorterDuff.Mode.DST_IN给我们带来的好处，本节我们继续来写例子练练手， 还记得8.3.2 绘图类实战示例给大家带来的拔掉美女衣服的实现吗？
>
> 
>
> 当时我们的实现方案是，将手指触碰区域附近的20*20个像素点设置为透明，效果图是这样的：
>
> 
>
>
> 不知道你有没有发现一个问题，我们擦美女衣服的时候，擦拭的时候都是方块的，但是我们画图板 画图的时候，划线都是很平滑的，有没有办法将两者结合起来，我们擦衣服时也是圆滑的呢？ 答案肯定是有的，就是使用Xfermode咯！本节我们使用另一个模式，**DST_OUT**模式！ 在不相交的地方绘制目标图



如果你忘记了某个模式或者连18种模式都没见过的话，那么请移步： Android基础入门教程——8.3.5 Paint API之—— Xfermode与PorterDuff详解(二) 另外，还是要贴下PorterDuff.Mode的效果图：



嗯，话不多说，开始本节内容~

1.要实现的效果图以及实现流程分析：

要实现的效果图：



> 
>
> 嗯，不知道你看了那个Gif图多少次了呢？不知道图中是否适合大家的口味，小猪 是从别人的APP上扒下来的，别问我番号或者留邮箱什么的，我什么都不知道~找番什么的，问群里的老司机——基神吧，好的，我们来分析下实现流程吧~
>
> - 我们来说说原理，其实就是两个Bitmap，一前一后，前面的是穿着衣服的，后面的是没穿衣服的，然后通过一个Path来记录用户绘制出来的图形，然后为我们的画笔设置DST_OUT的模式，那么 与Path重叠部分的DST(目标图)，就是穿着衣服的图，会变成透明！好哒，很简单！我们再慢慢细化！
> - 首先我们需要两个Bitmap，用来存储前后两张图片，这里我们让两个Bitmap都全屏！
> - 接着设置下画笔，圆角，笔宽，抗锯齿等！
> - 再接着定义一个画Path，即用户绘制区域的方法，设置Xfermode后画区域而已！
> - 然后重写onTouchEvent方法，这部分和之前的自定义画图板是一样的！
> - 最后重写onDraw()方法，先绘制背景图片，调用用户绘制区域的方法，再绘制前景图片！

可能看上去有点复杂，其实不然，代码超简单的说~

2.代码实现：

直接就一个自定义View——**StripMeiZi.java**

```
/**
 * Created by Jay on 2015/10/25 0025.
 */
public class StripMeiZi extends View{

    private Paint mPaint = new Paint();
    private Path mPath = new Path();
    private Canvas mCanvas;
    private Bitmap mBeforeBitmap;
    private Bitmap mBackBitmap;
    private int mLastX,mLastY;
    private int screenW, screenH; //屏幕宽高
    private Xfermode mXfermode = new PorterDuffXfermode(PorterDuff.

    public StripMeiZi(Context context) {
        this(context, null);
    }

    public StripMeiZi(Context context, AttributeSet attrs) {
        super(context, attrs);
        screenW = ScreenUtil.getScreenW(context);
        screenH = ScreenUtil.getScreenH(context);
        init();
    }

    public StripMeiZi(Context context, AttributeSet attrs, int defS
        super(context, attrs, defStyleAttr);
    }

    private void init() {
        //背后图片，这里让它全屏
        mBackBitmap = BitmapFactory.decodeResource(getResources(),
        mBackBitmap = Bitmap.createScaledBitmap(mBackBitmap, screen
        //前面的图片，并绘制到Canvas上
        mBeforeBitmap = Bitmap.createBitmap(screenW, screenH, Bitma
        mCanvas = new Canvas(mBeforeBitmap);
        mCanvas.drawBitmap(BitmapFactory.decodeResource(getResource
                R.mipmap.meizi_before), null, new RectF(0, 0, scree
        //画笔相关的设置
        mPaint.setAntiAlias(true);
        mPaint.setDither(true);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeJoin(Paint.Join.ROUND); // 圆角
        mPaint.setStrokeCap(Paint.Cap.ROUND); // 圆角
        mPaint.setStrokeWidth(80);    // 设置画笔宽
    }

    private void drawPath() {
        mPaint.setXfermode(mXfermode);
        mCanvas.drawPath(mPath, mPaint);
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(mBackBitmap, 0, 0, null);
        drawPath();
        canvas.drawBitmap(mBeforeBitmap, 0, 0, null);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int action = event.getAction();
        int x = (int) event.getX();
        int y = (int) event.getY();
        switch (action)
        {
            case MotionEvent.ACTION_DOWN:
                mLastX = x;
                mLastY = y;
                mPath.moveTo(mLastX, mLastY);
                break;
            case MotionEvent.ACTION_MOVE:

                int dx = Math.abs(x - mLastX);
                int dy = Math.abs(y - mLastY);

                if (dx > 3 || dy > 3)
                    mPath.lineTo(x, y);

                mLastX = x;
                mLastY = y;
                break;
        }
        invalidate();
        return true;
    }
}
```

布局代码**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.jay.xfermodedemo2.StripMeiZi
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

3.代码示例下载：

XfermodeDemo2.zip

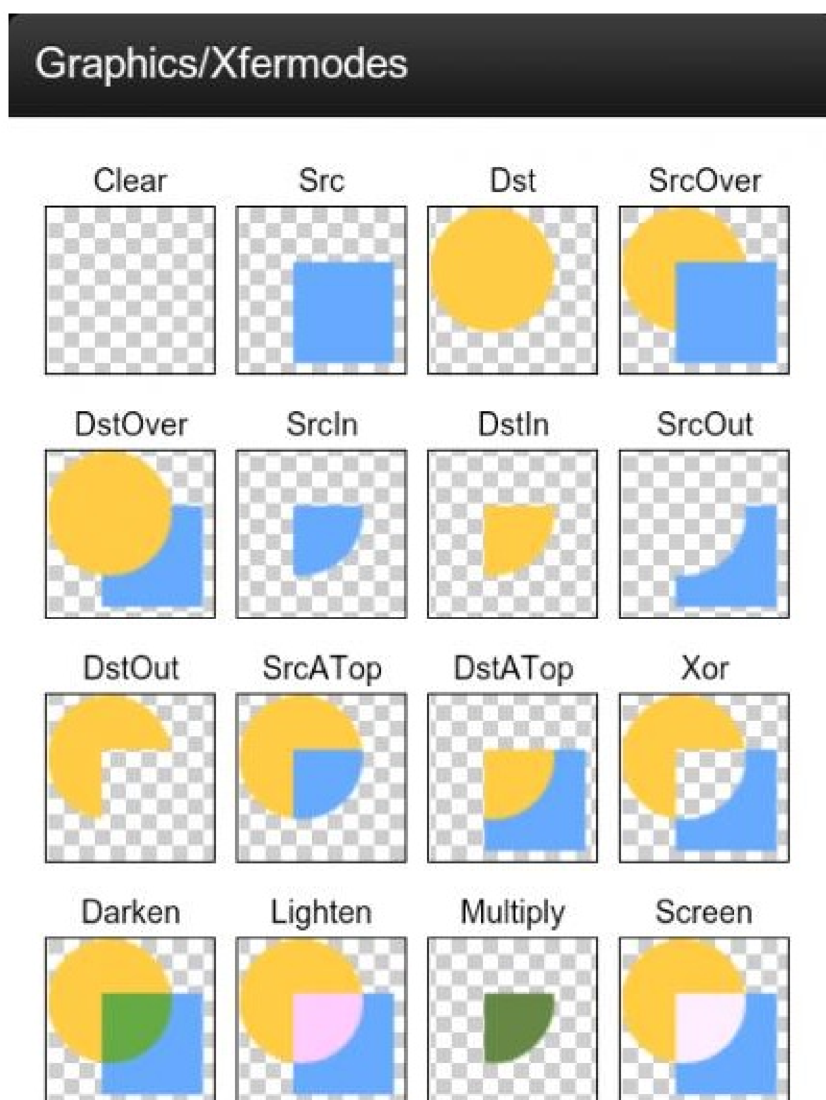
本节小结：

> 好的，本节我们写了Xfermode与PorterDuff的另一个实战例子——手撕美女衣服的Demo，相比起我们之前 那种撕美女衣服(让触摸点附近20*20的像素点变成透明)的方式斯文多了~代码也简单很多是吧，有没有 体会到Android图像混排Xfermode给我们带来的好处，或者对于自定义控件的重要性！嗯，还等什么， 打开你的IDE，把代码撸一遍，尝尝手撕美女衣服的快感吧~ 

8.3.8 Paint API之—— Xfermode与PorterDuff详解(五)

本节引言：

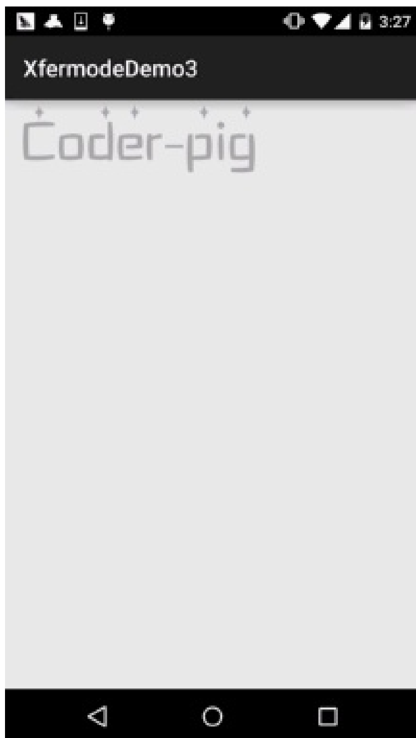
好的，上一节中，我们又写了一个关于Xfermode图片混排的例子——擦美女衣服的Demo，加上前面的 利用Xfermode来实现圆角或圆形ImageView，相信大家对Xfermode已经不再像以前那么陌生了，或者 说有点熟悉了，嗯，本节我们来写Xfermode的最后一个例子，通过Xfermode的ProterDuff.**SRC_IN** 模式来实现文字加载的效果！还是得贴下ProterDuff的模式图：



本节例子参考自：Android Paint之 setXfermode PorterDuffXfermode 讲解
嗯，话不多说，开始本节内容~

1.要实现的效果图以及实现流程分析：

要实现的效果图：



实现流程分析：

> **Step 1**.首先，一个文字图片(透明背景)
>
> **Step 2**.初始化画笔，背景图片(DST)，矩形Rect(SRC)
>
> **Step 3**.先保存图层，接着先绘制背景图，设置混排模式，然后绘制Rect，清除混排模式 接着回复保存的图层，最后修改下Rect区域高度，调用invalidate()让View重绘！

如果流程分析有点不懂，直接看代码，超简单~

2.代码实现：

首先是屏幕工具类，ScreenUtil.java，这里就不贴了，之前的几节中有贴过！ 然后是我们的自定义View类：**LoadTextView.java**：

```
/**
 * Created by Jay on 2015/10/26 0026.
 */
public class LoadTextView extends View {

    private PorterDuffXfermode mXfermode = new PorterDuffXfermode(P
    private Bitmap backBitmap;
    private Paint mPaint;
    private int mBitW, mBitH;
    private int mCurW, mCurH, mCurTop;
    private Rect mDynamicRect;
```

```
    public LoadTextView(Context context) {
        this(context, null);
    }
    public LoadTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mCurW = ScreenUtil.getScreenW(context);
        mCurH = ScreenUtil.getScreenH(context);
        init();
    }
    public LoadTextView(Context context, AttributeSet attrs, int de
        super(context, attrs, defStyleAttr);
    }

    private void init() {
        //画笔初始化：
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mPaint.setFilterBitmap(true);
        mPaint.setDither(true);
        mPaint.setColor(Color.RED);
        //背部图片的初始化
        backBitmap = BitmapFactory.decodeResource(getResources(), R
        mBitH = backBitmap.getHeight();
        mBitW = backBitmap.getWidth();
        //设置当前的高度
        mCurTop = mBitH;
        mDynamicRect = new Rect(0, mBitH, mBitW, mBitH);  //初始化原
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int saveLayerCount = canvas.saveLayer(0, 0, mCurW, mCurH, m
        canvas.drawBitmap(backBitmap, 0, 0, mPaint);// 绘制目标图
        mPaint.setXfermode(mXfermode);    //设置混排模式
        canvas.drawRect(mDynamicRect, mPaint);   //绘制源图
        mPaint.setXfermode(null);         //清除混排模式
        canvas.restoreToCount(saveLayerCount);    //恢复保存的图层
        // 改变Rect区域，假如
        mCurTop -= 2;
        if (mCurTop <= 0) {
            mCurTop = mBitH;
        }
        mDynamicRect.top = mCurTop;
        invalidate();    //重绘
    }

}
```

嗯，没有了，就上面这么点代码，就实现了如图所示的效果，是不是很简单咧~

要coder-pig字体的图片么，贴下~

3.本节代码示例下载：

XfermodeDemo3.zip

本节小结：

> 好的，本节我们又用PorterDuff的**SRC_IN**模式来写一个文字加载的效果，加上前面的：**DST_IN**模式实现圆形和圆角ImageView，以及**DST_OUT**模式来实现擦掉美女衣服，相信 大家对Xfermode的使用已经有眉目了，当然这些例子都是没有太大意义的，实际开发根本不会 用到，不过很便于大家理解~就好像练功夫，师傅领进门，修行靠自身！基础教程只是一个引导 而已，要真正掌握并学以致用还需靠你们自己，多阅读别人的优秀的代码，以及多动手！ 好的，就说这么多，谢谢~

8.3.9 Paint API之—— ColorFilter(颜色过滤器)(1/3)

本节引言：

> 上节我们学习了MaskFilter(面具)，用它的两个子类BlurMaskFilter弄了下模糊效果，EmbossMaskFilter 弄了下浮雕效果，而本节我们来学习的是另一个API——**ColorFilter**(颜色过滤器)，和MaskFilter一样， 我们并不直接使用该类，而是使用该类的三个子类：
>
> 颜色矩阵颜色过滤器：ColorMatrixColorFilter
>
> 光照色彩过滤器：LightingColorFilter
>
> 混排颜色过滤器滤器PorterDuffColorFilter
>
> 本节我们就来学习下第一个ColorMatrixColorFilter的使用吧，打开ColorMatrixColorFilter的文档，
>
> public class
>
> ColorMatrixColorFilter
>
> extends ColorFilter
>
> java.lang.Object
> ↳android.graphics.ColorFilter
> ↳android.graphics.ColorMatrixColorFilter
>
> Class Overview
>
> A color filter that transforms colors through a 4x5 color matrix. This filter can be used to change the saturation of pixels, convert from YUV to RGB, etc.
>
> See Also
>
> `ColorMatrix`
>
> Summary
>
> | Public Constructors |
> | --- |
> | ColorMatrixColorFilter(ColorMatrix matrix)
Create a color filter that transforms colors through a 4x5 color matrix. |
> | ColorMatrixColorFilter(float[] array)
Create a color filter that transforms colors through a 4x5 color matrix. |
>
> 大概说的是：通过一个4 x 5的颜色矩阵来变换颜色，可以修改像素的饱和度，将YUV转换成RGB等！ 而构造方法中的ColorMatrix就是颜色矩阵，也是我们学习的核心，下面听我一一道来！
>
> PS：ColorMatrix的API文档

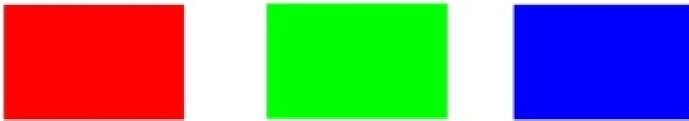
1.相关常识的普及：

RGBA模型：

> RGBA不知道你听过没，黄绿蓝知道了吧，光的三基色，而RAGB则是在此的基础上多了一个透明度！ **R(Red**红色)，**G(Green**绿色)，**B(Blue**蓝色)，**A(Alpha**透明度)；另外要和颜料的三 原色区分开来哦，最明显的区别就是颜料的三原色中用黄色替换了光三基色中的绿色！知道下就好， 有兴趣的可自行百度~

一些名词：

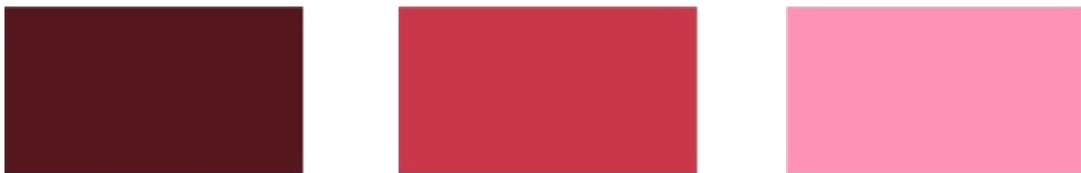
- 色调/色相——物体传递的颜色



- 饱和度——颜色的纯度，从0(灰)到100%(饱和)来进行描述

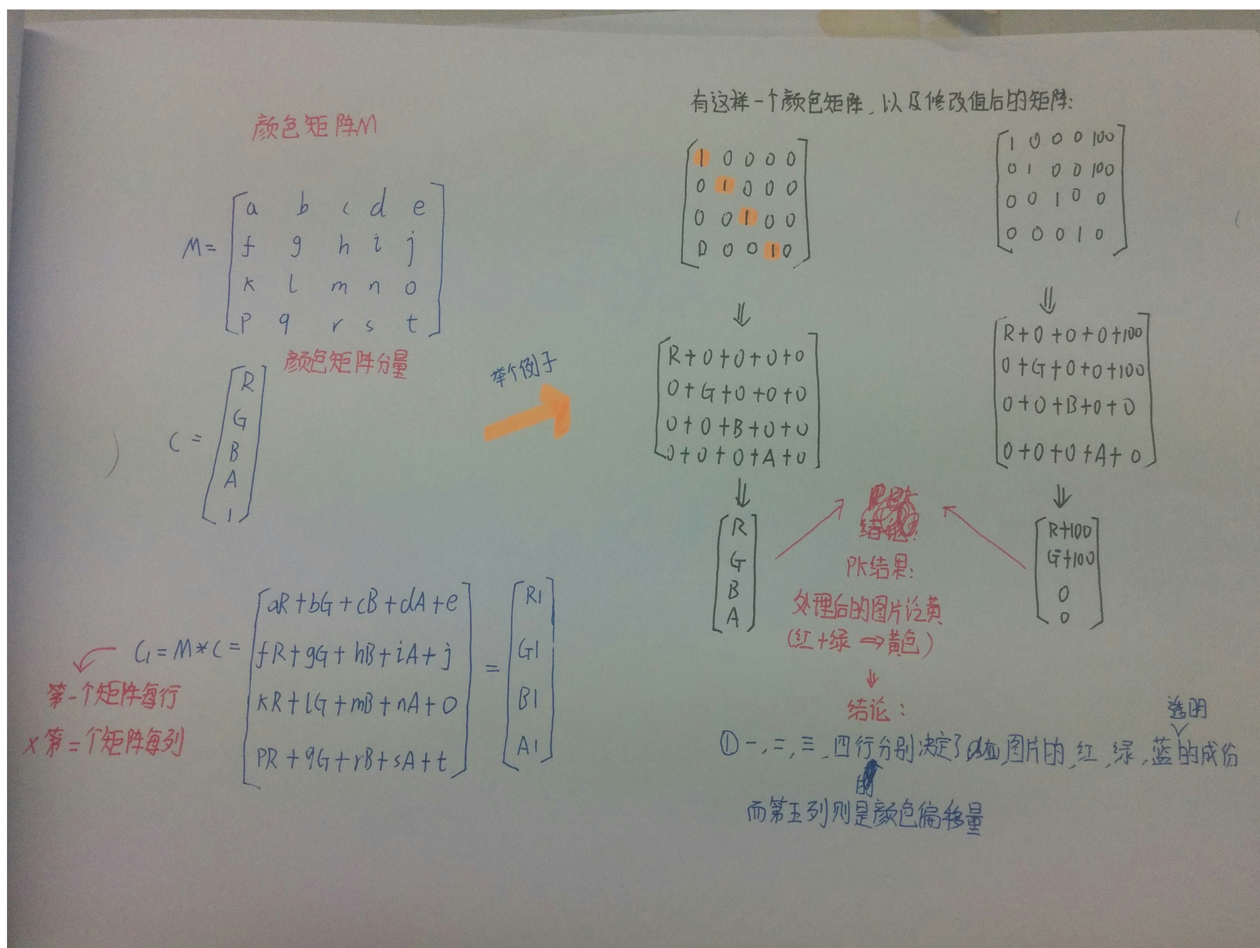


- 亮度/明度——颜色的相对明暗程度



2.ColorMatrix的解读

> 如题，颜色矩阵(4 * 5)，我们可以修改矩阵中的值，来实现黑白照，泛黄老照片，高对比度等效果！ 手撕颜色矩阵解释图如下：



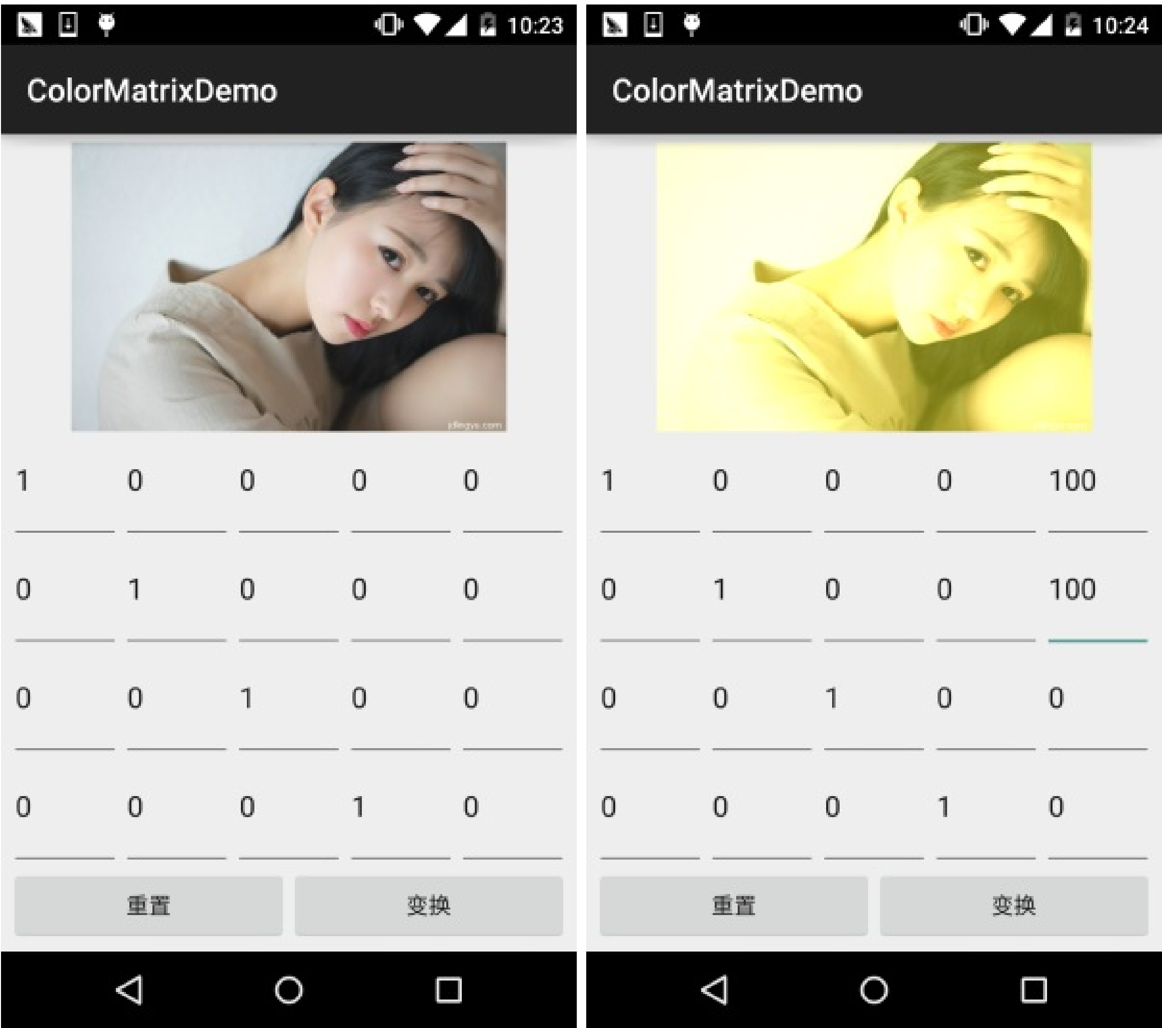
不知道你看懂上图没，如果你学过高数的话，肯定对此很熟悉，无非是矩阵的叉乘而已，没学过也没关系 计算方法就是右下角那个，拿颜色矩阵的每一行来 * 颜色矩阵分量的每一列！

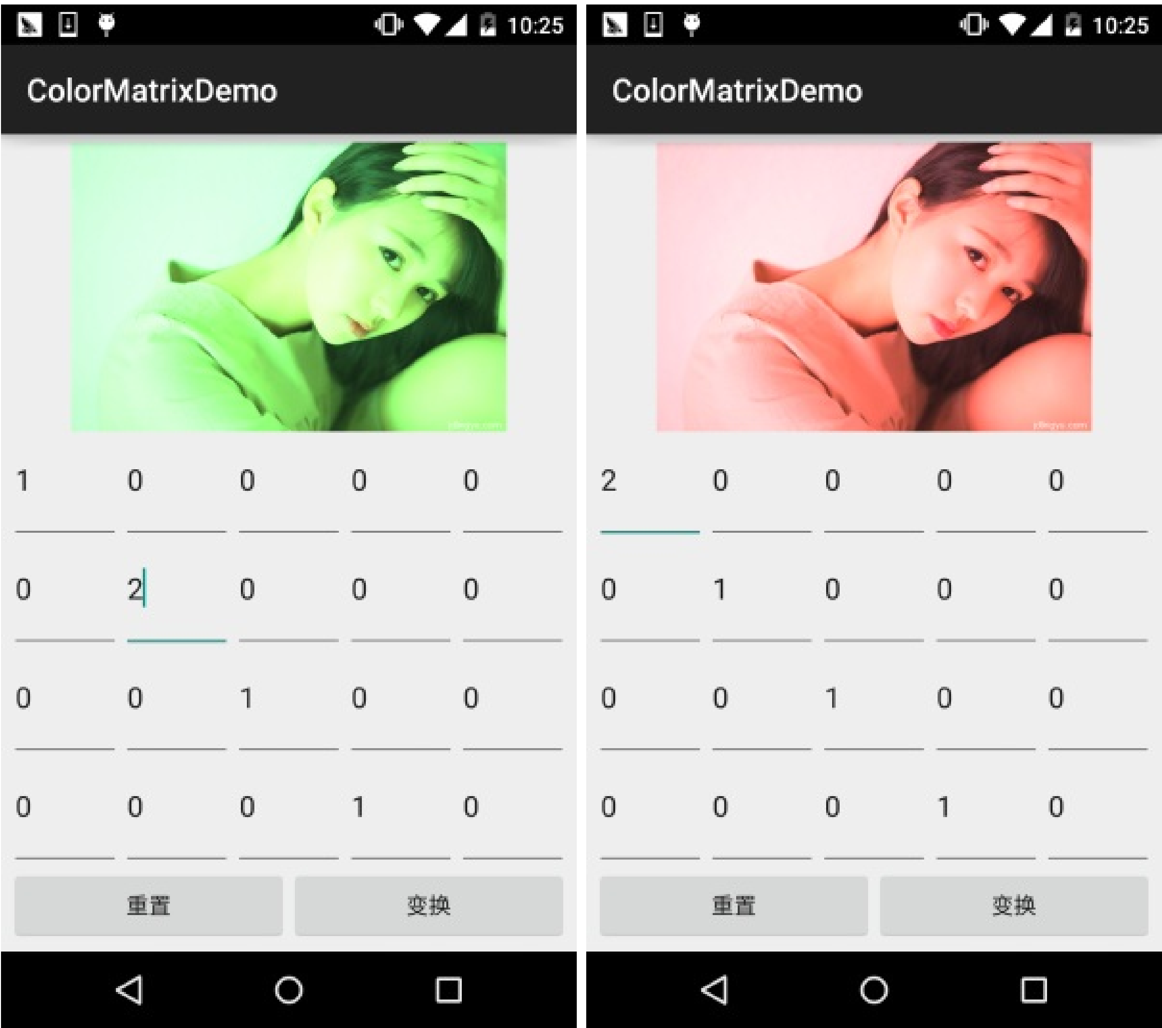
很典型的一个例子，处理前后的结果比较，我们还可以让某个颜色值 * 一个常数，比如让第三行(蓝) 乘以2，效果就变成泛蓝色了，当然，我们肯定要写代码来验证验证上面的结果！

3.写代码来验证ColorMatrix所起的作用

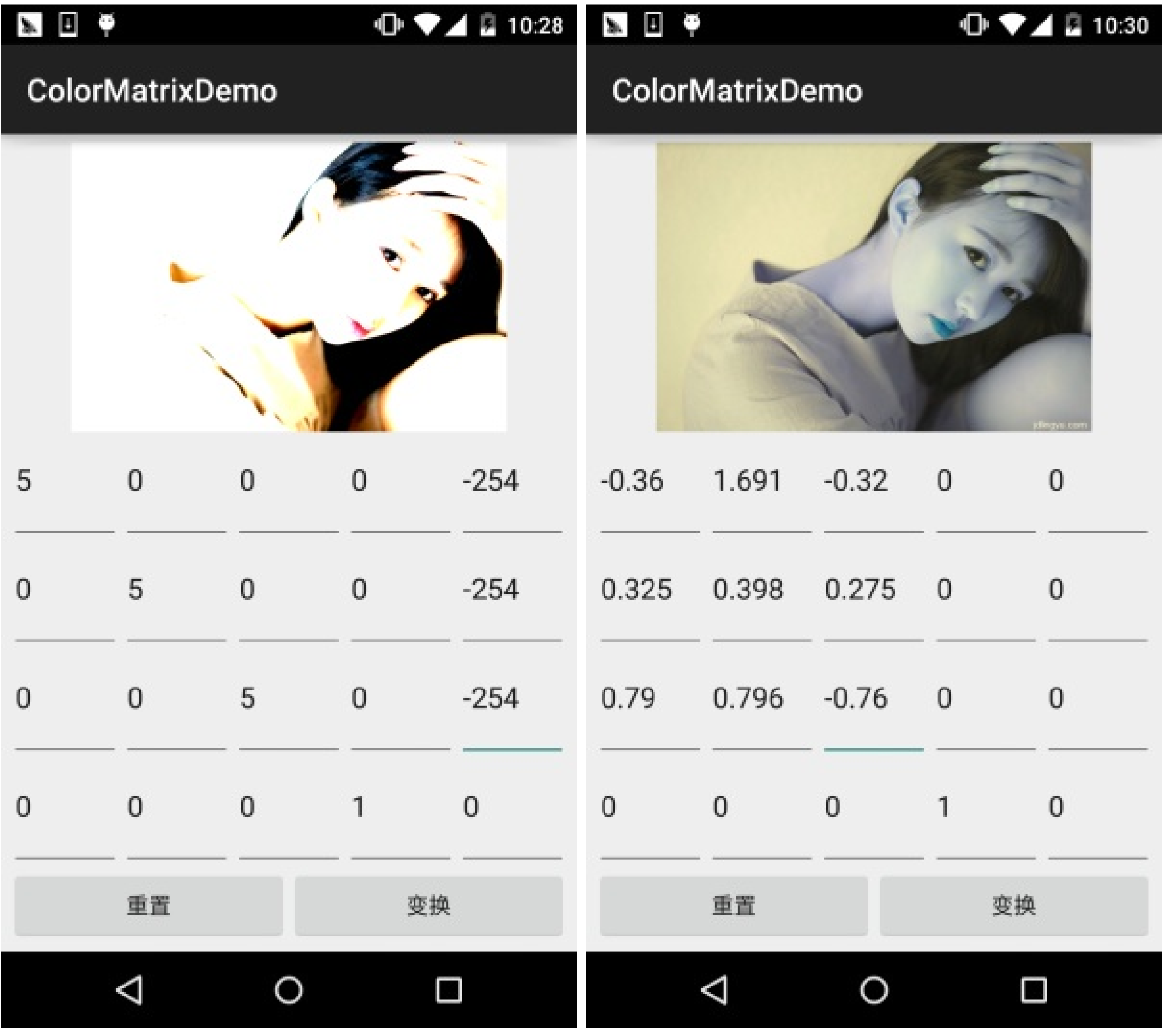
> 这里来写烂大街的例子，一个ImageView，4 * 5个EditText，一个重置按钮和一个生成按钮， 我们来看下效果图：

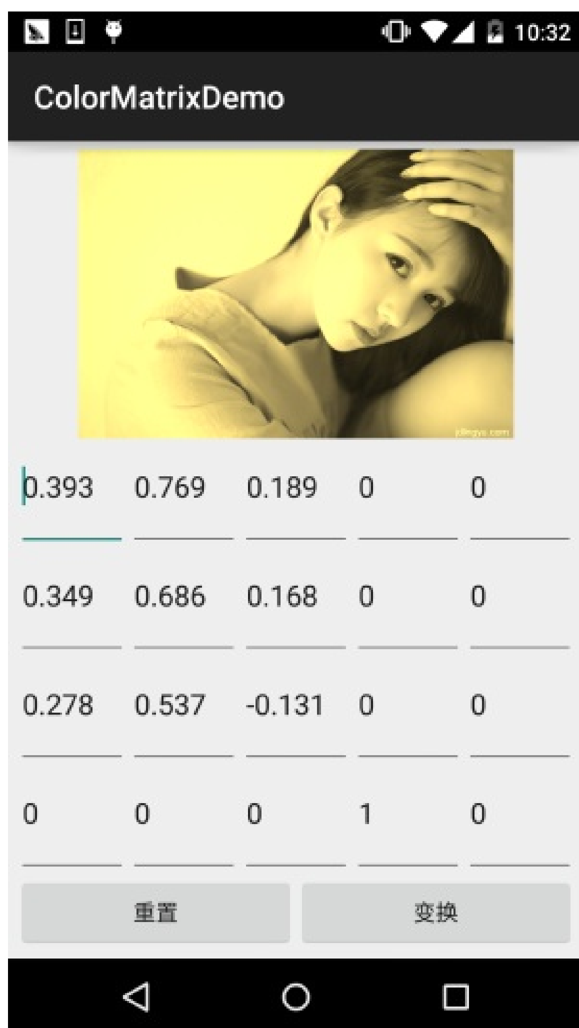
依次是原图，泛黄，泛绿，泛红，高对比度，色相变换，以及黄色复古


10:23
ColorMatrixDemo
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
重置
变换
10:24
ColorMatrixDemo
1 0 0 0 100
0 1 0 0 100
0 0 1 0 0
0 0 0 1 0
重置
变换



10:25
ColorMatrixDemo
1 0 0 0 0
0 2 0 0 0
0 0 1 0 0
0 0 0 1 0
重置
变换
10:25
ColorMatrixDemo
2 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
重置
变换


10:28
ColorMatrixDemo
5 0 0 0 -254
0 5 0 0 -254
0 0 5 0 -254
0 0 0 1 0
重置
变换
10:30
ColorMatrixDemo
-0.36 1.691 -0.32 0 0
0.325 0.398 0.275 0 0
0.79 0.796 -0.76 0 0
0 0 0 1 0
重置
变换



接下来我们来写代码，完成上述的效果： 代码实现：

首先是布局文件**activity_main.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <ImageView
        android:id="@+id/img_show"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2" />

    <GridLayout
        android:id="@+id/gp_matrix"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:columnCount="5"
        android:rowCount="4"></GridLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btn_reset"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="重置" />
        <Button
            android:id="@+id/btn_Change"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="变换" />
    </LinearLayout>
</LinearLayout>
```

接着是**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity implements View

    private ImageView img_show;
    private GridLayout gp_matrix;
    private Button btn_reset;
    private Button btn_Change;
    private Bitmap mBitmap;
    private int mEtWidth, mEtHeight;
```

```
    private EditText[] mEts = new EditText[20];
    private float[] mColorMatrix = new float[20];
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mContext = MainActivity.this;
        bindViews();

        gp_matrix.post(new Runnable() {
            @Override
            public void run() {
                mEtWidth = gp_matrix.getWidth() / 5;
                mEtHeight = gp_matrix.getHeight() / 4;
                //添加5 * 4个EditText
                for (int i = 0; i < 20; i++) {
                    EditText editText = new EditText(mContext);
                    mEts[i] = editText;
                    gp_matrix.addView(editText, mEtWidth, mEtHeight
                }
                initMatrix();
            }
        });
    }

    private void bindViews() {
        img_show = (ImageView) findViewById(R.id.img_show);
        gp_matrix = (GridLayout) findViewById(R.id.gp_matrix);
        btn_reset = (Button) findViewById(R.id.btn_reset);
        btn_Change = (Button) findViewById(R.id.btn_Change);

        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
        img_show.setImageBitmap(mBitmap);

        btn_reset.setOnClickListener(this);
        btn_Change.setOnClickListener(this);
    }

    //定义一个初始化颜色矩阵的方法
    private void initMatrix() {
        for (int i = 0; i < 20; i++) {
            if (i % 6 == 0) {
                mEts[i].setText(String.valueOf(1));
            } else {
                mEts[i].setText(String.valueOf(0));
            }
        }
    }

    //定义一个获取矩阵值得方法
    private void getMatrix() {
```

```
        for (int i = 0; i < 20; i++) {
            mColorMatrix[i] = Float.valueOf(mEts[i].getText().toStr
        }
    }

    //根据颜色矩阵的值来处理图片
    private void setImageMatrix() {
        Bitmap bmp = Bitmap.createBitmap(mBitmap.getWidth(), mBitma
                Bitmap.Config.ARGB_8888);
        android.graphics.ColorMatrix colorMatrix = new android.grap
        colorMatrix.set(mColorMatrix);

        Canvas canvas = new Canvas(bmp);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColorFilter(new ColorMatrixColorFilter(colorMatrix
        canvas.drawBitmap(mBitmap, 0, 0, paint);
        img_show.setImageBitmap(bmp);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_Change:
                getMatrix();
                setImageMatrix();
                break;
            case R.id.btn_reset:
                initMatrix();
                getMatrix();
                setImageMatrix();
                break;
        }
    }
}
```

代码非常的简单，就加载布局，然后往GridLayout里面塞 5 * 4 个EditText，这里用post()方法是为了保证GridLayout加载完毕后才去获取长宽，不然在获取GridLayout长 宽的时候可是获取不到值的！接着定义了三个方法，初始矩阵，获取矩阵值，以及根据 矩阵值来处理图片~是不是很简单咧~

不过到这里你可能有一点疑问：

> " 难道处理图像我们只能这样修改颜色矩阵么？次次都这样肯定很麻烦，谁会去记矩阵 里的应该填的值？有没有简单一点处理图片的方法？ "
>
> 答：肯定是有的，我们可以看回文档，我们可以发现几个很常用的方法：**setRotate**(int axis, float degrees)：设置色调
>
> **setSaturation**(float sat)：设置饱和度
>
> **setScale**(float rScale, float gScale, float bScale, float aScale)：设置亮度
>
> 下面我们写个例子来试下这个三个方法！

4.使用ColorMatrix的三个方法处理图像

运行效果图：



代码实现：

首先我们来编写一个图片处理的工具类，我们传入Bitmap，色相，饱和度以及亮度，处理后，返回 处理后的图片：**ImageHelper.java**：

```
/**
 * Created by Jay on 2015/10/28 0028.
 */
public class ImageHelper {
    /**
     * 该方法用来处理图像，根据色调，饱和度，亮度来调节
     *
     * @param bm:要处理的图像
     * @param hue:色调
     * @param saturation:饱和度
     * @param lum:亮度
     *
     */
    public static Bitmap handleImageEffect(Bitmap bm, float hue, fl
        Bitmap bmp = Bitmap.createBitmap(bm.getWidth(), bm.getHeigh
        Canvas canvas = new Canvas(bmp);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);

        ColorMatrix hueMatrix = new ColorMatrix();
        hueMatrix.setRotate(0, hue);    //0代表R，红色
        hueMatrix.setRotate(1, hue);    //1代表G，绿色
        hueMatrix.setRotate(2, hue);    //2代表B，蓝色

        ColorMatrix saturationMatrix = new ColorMatrix();
        saturationMatrix.setSaturation(saturation);

        ColorMatrix lumMatrix = new ColorMatrix();
        lumMatrix.setScale(lum, lum, lum, 1);

        ColorMatrix imageMatrix = new ColorMatrix();
        imageMatrix.postConcat(hueMatrix);
        imageMatrix.postConcat(saturationMatrix);
        imageMatrix.postConcat(lumMatrix);

        paint.setColorFilter(new ColorMatrixColorFilter(imageMatrix
        canvas.drawBitmap(bm, 0, 0, paint);

        return bmp;
    }
}
```

接下来我们把布局也撸出来，**activity_main.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
```

```
    <ImageView
        android:id="@+id/img_meizi"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="24dp"
        android:layout_marginTop="24dp" />

    <TextView
        android:id="@+id/txt_hue"
        android:layout_width="wrap_content"
        android:layout_height="32dp"
        android:layout_below="@id/img_meizi"
        android:gravity="center"
        android:text="色调    :"
        android:textSize="18sp" />

    <SeekBar
        android:id="@+id/sb_hue"
        android:layout_width="match_parent"
        android:layout_height="32dp"
        android:layout_below="@id/img_meizi"
        android:layout_toRightOf="@id/txt_hue" />

    <TextView
        android:id="@+id/txt_saturation"
        android:layout_width="wrap_content"
        android:layout_height="32dp"
        android:layout_below="@id/txt_hue"
        android:gravity="center"
        android:text="饱和度:"
        android:textSize="18sp" />

    <SeekBar
        android:id="@+id/sb_saturation"
        android:layout_width="match_parent"
        android:layout_height="32dp"
        android:layout_below="@id/sb_hue"
        android:layout_toRightOf="@id/txt_saturation" />

    <TextView
        android:id="@+id/txt_lun"
        android:layout_width="wrap_content"
        android:layout_height="32dp"
        android:layout_below="@id/txt_saturation"
        android:gravity="center"
        android:text="亮度    :"
        android:textSize="18sp" />

    <SeekBar
        android:id="@+id/sb_lum"
        android:layout_width="match_parent"
        android:layout_height="32dp"
```

```
        android:layout_below="@id/sb_saturation"
        android:layout_toRightOf="@id/txt_lun" />

</RelativeLayout>
```

最后是我们的**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity implements Seek

    private ImageView img_meizi;
    private SeekBar sb_hue;
    private SeekBar sb_saturation;
    private SeekBar sb_lum;
    private final static int MAX_VALUE = 255;
    private final static int MID_VALUE = 127;
    private float mHue = 0.0f;
    private float mStauration = 1.0f;
    private float mLum = 1.0f;
    private Bitmap mBitmap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
        bindViews();
    }

    private void bindViews() {
        img_meizi = (ImageView) findViewById(R.id.img_meizi);
        sb_hue = (SeekBar) findViewById(R.id.sb_hue);
        sb_saturation = (SeekBar) findViewById(R.id.sb_saturation);
        sb_lum = (SeekBar) findViewById(R.id.sb_lum);

        img_meizi.setImageBitmap(mBitmap);
        sb_hue.setMax(MAX_VALUE);
        sb_hue.setProgress(MID_VALUE);
        sb_saturation.setMax(MAX_VALUE);
        sb_saturation.setProgress(MID_VALUE);
        sb_lum.setMax(MAX_VALUE);
        sb_lum.setProgress(MID_VALUE);

        sb_hue.setOnSeekBarChangeListener(this);
        sb_saturation.setOnSeekBarChangeListener(this);
        sb_lum.setOnSeekBarChangeListener(this);

    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, bo
```

```
        switch (seekBar.getId()) {
            case R.id.sb_hue:
                mHue = (progress - MID_VALUE) * 1.0F / MID_VALUE *
                break;
            case R.id.sb_saturation:
                mStauration = progress * 1.0F / MID_VALUE;
                break;
            case R.id.sb_lum:
                mLum = progress * 1.0F / MID_VALUE;
                break;
        }
        img_meizi.setImageBitmap(ImageHelper.handleImageEffect(mBit
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}
}
```

代码同样很简单，这里就不讲解了~

5.本节代码示例下载：

ColorMatrixDemo.zip

ColorMatrixDemo2.zip

本节小结：

> 好的，本节跟大家介绍了**ColorFilter**中的第一个**ColorMatrixColorFilter**，颜色矩阵过滤器 其实核心还是ColorMatrix，我们通过该类处理图片可以自己设置4*5矩阵的值，又或者直接调用 ColorMatrix给我们提供的设置色调，饱和度，亮度的方法！图像处理无非就这样，还有一种是修改 像素点形式的，后面也会讲，本节内容参考自——医生(徐宜生)的慕客网视频：Android图像处理-打造美图秀秀从它开始，不想看文字的可以看视频，讲得还是蛮赞的~

8.3.10 Paint API之—— ColorFilter(颜色过滤器)(2-3)

本节引言：

> 上一节中我们讲解了Android中Paint **API**中的**ColorFilter**(颜色过滤器)的第一个子类： **ColorMatrixColorFilter**(颜色矩阵颜色过滤器)，相信又开阔了大家的Android图像处理视野， 而本节我们来研究它的第二个子类：**LightingColorFilter**(光照色彩颜色过滤器)，先上一发 官方API文档：LightingColorFilter， 文档里的东西也不多， 关键的在这里：
>
> LightingColorFilter
>
> extends ColorFilter
>
> java.lang.Object
> └android.graphics.ColorFilter
> 　└android.graphics.LightingColorFilter
>
> Class Overview
>
> A color filter that can be used to simulate simple lighting effects. A `LightingColorFilter` is defined by two parameters, one used color (called `colorAdd`). The alpha channel is left untouched by this color filter. Given a source color RGB, the resulting R'G'B' colo
>
> ```
> R' = R * colorMultiply.R + colorAdd.R
> G' = G * colorMultiply.G + colorAdd.G
> B' = B * colorMultiply.B + colorAdd.B
> ```
>
> The result is pinned to the `[0..255]` range for each channel.
>
> Summary
>
> | Public Constructors |
> |---|
> | LightingColorFilter(int mul, int add)
Create a colorfilter that multiplies the RGB channels by one color, and then adds a second color. |
>
> 大概意思就是：一个颜色过滤器， 可以用来模拟简单的灯光效果， 构造方法的参数有两个， 一个 用来乘以原图的RPG值， 一个添加到前面得出的结果上！其实计算方法无非： **(RGB值 * mul + Add) % 255**， 从而得到新的RPG值， 这里的%是求余， 另外， 整个过程中Alpha不 参与改变！下面我们写个示例来验证验证！

1.代码示例：

运行效果图：



实现代码：

先是一个简单的布局：**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/img_meizi"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:src="@mipmap/img_meizi" />

    <EditText
        android:id="@+id/edit_mul"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/img_meizi"
        android:text="0" />

    <EditText
        android:id="@+id/edit_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/edit_mul"
        android:text="0" />

    <Button
        android:id="@+id/btn_change"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@id/img_meizi"
        android:layout_below="@id/img_meizi"
        android:text="变化" />

</RelativeLayout>
```

接着是我们的**MainActiivty.java**，同样很简单：

```
public class MainActivity extends AppCompatActivity implements View

    private ImageView img_meizi;
    private EditText edit_mul;
    private EditText edit_add;
    private Button btn_change;
    private Bitmap mBitmap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
        bindViews();
    }

    private void bindViews() {
        img_meizi = (ImageView) findViewById(R.id.img_meizi);
        edit_mul = (EditText) findViewById(R.id.edit_mul);
        edit_add = (EditText) findViewById(R.id.edit_add);
        btn_change = (Button) findViewById(R.id.btn_change);

        btn_change.setOnClickListener(this);

    }

    private Bitmap ProcessImage(Bitmap bp,int mul,int add){
        Bitmap bitmap = Bitmap.createBitmap(bp.getWidth(),bp.getHei
        Canvas canvas = new Canvas(bitmap);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColorFilter(new LightingColorFilter(mul,add));
        canvas.drawBitmap(bp,0,0,paint);
        return bitmap;
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_change:
                int mul = Integer.parseInt(edit_mul.getText().toStr
                int add = Integer.parseInt(edit_add.getText().toStr
                img_meizi.setImageBitmap(ProcessImage(mBitmap,mul,a
                break;
        }
    }
}
```

好了，LightingColorFilter的使用演示完毕~

3.本节代码下载

LightingColorFilterDemo.zip

本节小结：

> 嗯，本节演示了一下LightingColorFilter的一个基本用法，用来模拟简单的灯光效果， 实现简单的图片处理效果，好的，本节就到这里，谢谢~

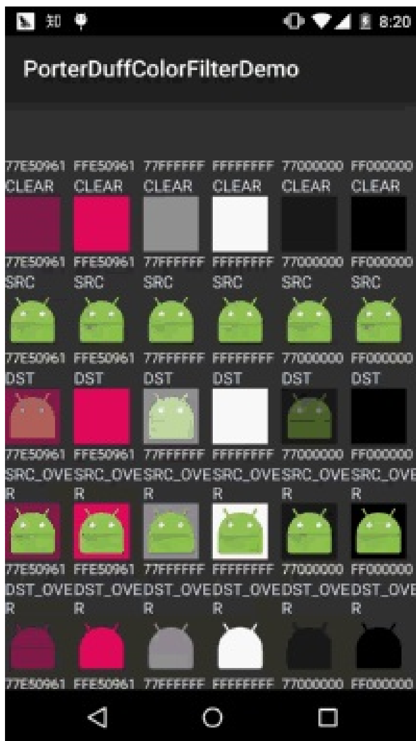
8.3.11 Paint API之—— ColorFilter(颜色过滤器)(3-3)

本节引言：

> 嗯，本来说好今天不写的，还是写吧，毕竟难得空闲哈~，本节给大家带来的是 ColorFilter的第三个子类：**PorterDuffColorFilter**，看到**PorterDuff**大家一定不会 陌生吧，假如你看过前面的 Android基础入门教程——8.3.5 Paint API之—— Xfermode与PorterDuff详解(二) 其实效果都是一样的，只是这里用的是颜色，而且直接设置就好，下面我们来写个简单的 例子，我们取6种不同的颜色，对18种模式进行测试！ 官方API文档：PorterDuffColorFilter 我们可以看到关键也是在于他的构造方法：
>
> public class
>
> PorterDuffColorFilter
>
> extends ColorFilter
>
> java.lang.Object
> ↳android.graphics.ColorFilter
> ↳android.graphics.PorterDuffColorFilter
>
> ### Class Overview
>
> A color filter that can be used to tint the source pixels using a single color and a specific `Porter-Duff composite mode`.
>
> ### Summary
>
> | Public Constructors |
> | --- |
> | PorterDuffColorFilter(int color, PorterDuff.Mode mode)
Create a color filter that uses the specified color and Porter-Duff mode. |
>
> 前面是颜色，后面是模式~，来来来，写例子：

1.测试代码示例：

运行效果图：



代码实现：

这里的话我们用一个GridView来装他们，我们先来写下每个item的布局：**view_item.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/img_show"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:id="@+id/tv_color"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:textSize="12sp"
        android:text="颜色"
        android:textColor="#FFFFFFFF" />

    <TextView
        android:id="@+id/tv_mode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFD9ECFF"
        android:text="模式"/>

</LinearLayout>
```

接着我们编写一个POJO业务类：**Data.java**：

```
/**
 * Created by Jay on 2015/10/29 0029.
 */
public class Data {
    private int color;
    private PorterDuff.Mode mode;

    public Data() {
    }

    public Data(int color, PorterDuff.Mode mode) {
        this.color = color;
        this.mode = mode;
    }

    public int getColor() {
        return color;
    }

    public PorterDuff.Mode getMode() {
        return mode;
    }

    public void setColor(int color) {
        this.color = color;
    }

    public void setMode(PorterDuff.Mode mode) {
        this.mode = mode;
    }
}
```

至于Adapter类的话我们用回以前写的可复用的自定义BaseAdapter类，这里就不贴了，不过要加 多个方法：

```
/**
 * 设置ColorFilter
 * */
public ViewHolder setColorFilter(int id,int color,PorterDuff.Mode m
    View view = getView(id);
    if (view instanceof ImageView) {
        ((ImageView) view).setColorFilter(color,mode);
    }
    return this;
}
```

接着是我们的主布局文件:**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <GridView
        android:id="@+id/gd_show"
        android:background="#FF333333"
        android:numColumns="6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

最后是我们的**MainActivity.java**类，填充数据，设置Adapter，非常简单：

```
public class MainActivity extends AppCompatActivity {

    private GridView gd_show;
    private ArrayList<Data> items = null;
    private MyAdapter<Data> myAdapter = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        gd_show = (GridView) findViewById(R.id.gd_show);

        //填充数据，遍历Mode模式：
        items = new ArrayList<Data>();
        for (PorterDuff.Mode mode : PorterDuff.Mode.class.getEnumCo
            items.add(new Data(0x77E50961, mode));
            items.add(new Data(0xFFE50961, mode));
            items.add(new Data(0x77FFFFFF, mode));
            items.add(new Data(0xFFFFFFFF, mode));
            items.add(new Data(0x77000000, mode));
            items.add(new Data(0xFF000000, mode));
        }
        myAdapter = new MyAdapter<Data>(items, R.layout.view_item)
            @Override
            public void bindView(ViewHolder holder, Data obj) {
                holder.setColorFilter(R.id.img_show, obj.getColor(
                holder.setText(R.id.tv_color, String.format("%08X",
                holder.setText(R.id.tv_mode, obj.getMode().toStrin
            }
        };
        gd_show.setAdapter(myAdapter);
    }
}
```

上面的动图可能太快，有时读者相查下，这里分开图截，因为没找到好用的截全屏工具， 所以这里只能分段截...


77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
CLEAR CLEAR CLEAR CLEAR CLEAR CLEAR
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
SRC SRC SRC SRC SRC SRC
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
DST DST DST DST DST DST
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
SRC_OVER SRC_OVER SRC_OVER SRC_OVER SRC_OVER SRC_OVER
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
DST_OVER DST_OVER DST_OVER DST_OVER DST_OVER DST_OVER
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
SRC_IN SRC_IN SRC_IN SRC_IN SRC_IN SRC_IN
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
DST_IN DST_IN DST_IN DST_IN DST_IN DST_IN
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
SRC_OUT SRC_OUT SRC_OUT SRC_OUT SRC_OUT SRC_OUT
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
DST_OUT DST_OUT DST_OUT DST_OUT DST_OUT DST_OUT
77E50961 FFE50961 77FFFFFF FFFFFFFF 77000000 FF000000
SRC_ATOP SRC_ATOP SRC_ATOP SRC_ATOP SRC_ATOP SRC_ATOP



2.本节示例代码下载：

PorterDuffColorFilterDemo2.zip

本节小结：

> 本节非常简短，API文档里就那么个用法，这里也把18种情况也列举出来了，相信 会对大家学习图像混排带来帮助~谢谢，今天请了一天假，会学校又感受了下学生的 感觉，去了一趟图书馆，看了一大波的美女，然后心情就nice了，决定还是暂时先 在这个公司好好滴做一个实习生，换了环境不一定能改变什么，先从改变自己开始吧~
>
> 
>
>
> PS：例子摘自Github：ColorFilterTest

8.3.12 Paint API之—— PathEffect(路径效果)

本节引言：

> 本节继续来学习Paint的API——PathEffect(路径效果)，我们把画笔的sytle设置为Stroke，可以 绘制一个个由线构成的图形，而这些线偶尔会显得单调是吧，比如你想把这些先改成虚线，又 或者想让路径的转角变得圆滑等，那你就可以考虑使用这个PathEffect来实现了！
>
> 官方API文档：PathEffect 进去看文档，可以发现这个PathEffect和我们前面学的MaskFilter(面具)与ColorFilter(颜色 过滤器)一样，几乎没有可用的方法：
>
> public class
>
> PathEffect
>
> extends Object
>
> java.lang.Object
> ↳android.graphics.PathEffect
>
> ▸ Known Direct Subclasses
>
> ComposePathEffect, CornerPathEffect, DashPathEffect, DiscretePathEffect, PathDashPathEffect, SumPathEffect
>
> Class Overview
>
> PathEffect is the base class for objects in the Paint that affect the geometry of a drawing primitive before it is tra
>
> Summary
>
> | Public Constructors |
> | --- |
> | PathEffect() |
>
> 我们一般使用的是他的六个子类：
>
> - ComposePathEffect
> - CornerPathEffect
> - DashPathEffect
> - DiscretePathEffect
> - PathDashPathEffect
> - SumPathEffect
>
> 下面我们依次对他们的作用，以及构造方法进行分析！

1.子类作用与构造方法参数分析：

1)CornerPathEffect

CornerPathEffect(float radius)

> 将Path的各个连接线段之间的夹角用一种更平滑的方式连接，类似于圆弧与切线的效果。 radius则是指定圆弧的半径！

2)DashPathEffect

DashPathEffect(float[] intervals, float phase)

> 将Path的线段虚线化，intervals为虚线的ON和OFF的数组，数组中元素数目需要 >= 2； 而phase则为绘制时的偏移量！

3)DiscretePathEffect

DiscretePathEffect(float segmentLength, float deviation)

> 打散Path的线段，使得在原来路径的基础上发生打散效果。 segmentLength指定最大的段长，deviation则为绘制时的偏离量。

4)PathDashPathEffect

PathDashPathEffect(Path shape, float advance, float phase, PathDashPathEffect.Style style)

> 作用是使用Path图形来填充当前的路径，shape指的填充图形，advance是每个图形间的间隔， style则是该类自由的枚举值，有三种情况：**ROTATE**、**MORPH**和**TRANSLATE**。
>
> - ROTATE情况下：线段连接处的图形转换以旋转到与下一段移动方向相一致的角度进行连接
> - MORPH情况下：图形会以发生拉伸或压缩等变形的情况与下一段相连接
> - TRANSLATE情况下：图形会以位置平移的方式与下一段相连接

5)ComposePathEffect

ComposePathEffect(PathEffect outerpe, PathEffect innerpe)

> 作用是：组合效果，会首先将innerpe变现出来，接着在innerpe的基础上来增加outerpe效果！

6)SumPathEffect

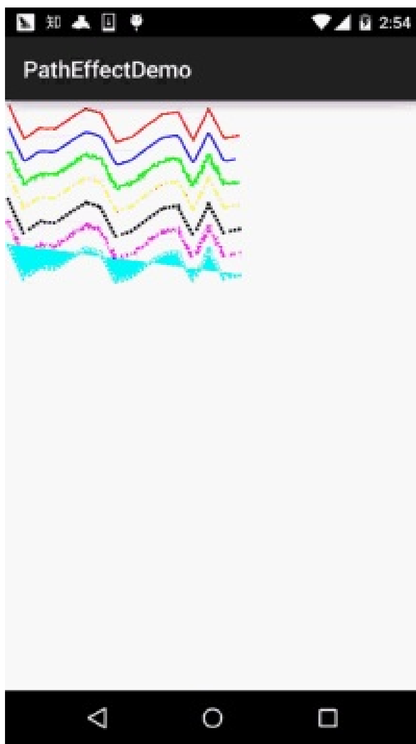
SumPathEffect(PathEffect first, PathEffect second)

> 作用是：叠加效果，和ComposePathEffect不同，在表现时会将两个参数的效果都独立的表现出来， 接着将两个效果简单的重叠在一起显示出来！

2.写代码来验证各自的效果

多说无益，写代码最实际，我们写下代码来试试这几个子类各自所起的效果！

运行效果图：



实现代码：

我们自己来写一个View，里面的线移动的效果是phase增加造成的，每次 + 2， 然后invalidate重绘而已，所以别惊讶！**PathEffectView.java**:

```
/**
 * Created by Jay on 2015/10/30 0030.
 */
public class PathEffectView extends View {

    private Paint mPaint;
    private Path mPath;
    private float phase = 0;
    private PathEffect[] effects = new PathEffect[7];
    private int[] colors;

    public PathEffectView(Context context) {
        this(context, null);
    }

    public PathEffectView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }
```

```
    public PathEffectView(Context context, AttributeSet attrs, int
        super(context, attrs, defStyleAttr);
    }

    //初始化画笔
    private void init() {
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG); //抗锯齿
        mPaint.setStyle(Paint.Style.STROKE);       //绘画风格:空心
        mPaint.setStrokeWidth(5);                  //笔触粗细
        mPath = new Path();
        mPath.moveTo(0, 0);
        for (int i = 1; i <= 15; i++) {
            // 生成15个点，随机生成它们的坐标，并将它们连成一条Path
            mPath.lineTo(i * 40, (float) Math.random() * 100);
        }
        // 初始化7个颜色
        colors = new int[] { Color.RED, Color.BLUE, Color.GREEN,
                Color.YELLOW, Color.BLACK, Color.MAGENTA, Color.CYA
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawColor(Color.WHITE);
        //初始化其中路径效果：
        effects[0] = null;                                          //无效
        effects[1] = new CornerPathEffect(10);                      //Cor
        effects[2] = new DiscretePathEffect(3.0f, 5.0f);            //Dis
        effects[3] = new DashPathEffect(new float[] { 20, 10, 5, 10
        Path p = new Path();
        p.addRect(0, 0, 8, 8, Path.Direction.CCW);
        effects[4] = new PathDashPathEffect(p, 12, phase,
                PathDashPathEffect.Style.ROTATE);                   //Pat
        effects[5] = new ComposePathEffect(effects[2], effects[4]);
        effects[6] = new SumPathEffect(effects[2], effects[4]);   /
        // 将画布移动到(10,10)处开始绘制
        canvas.translate(10, 10);
        // 依次使用7中不同的路径效果、7中不同的颜色来绘制路径
        for (int i = 0; i < effects.length; i++) {
            mPaint.setPathEffect(effects[i]);
            mPaint.setColor(colors[i]);
            canvas.drawPath(mPath, mPaint);
            canvas.translate(0, 60);
        }
        // 改变phase值，形成动画效果
        phase += 2;
        invalidate();
    }
}
```

好的，代码的注释已经非常清楚了，这里也不唠叨了~

3.本节示例代码下载：

PathEffectDemo.zip

本节小结

> 本节没什么难的东西，就介绍了PathEffect的六个子类所起的作用，只需调用setPathEffect 方法应用到Paint对象中，非常简单~嗯，就说这么多，谢谢~

8.3.13 Paint API之—— Shader(图像渲染)

1.构造方法详解

1)BitmapShader(图像渲染)

BitmapShader(Bitmap bitmap, Shader.TileMode tileX, Shader.TileMode tileY)

> 使用一张位图作为纹理来对某一区域进行填充，参数依次：
>
> - **bitmap**：用来作为填充的位图；
> - **tileX**：X轴方向上位图的衔接形式；
> - **tileY**：Y轴方向上位图的衔接形式；
>
> 而这个Shader.TileMode有三种：
>
> - **CLAMP**就是如果渲染器超出原始边界范围，则会复制边缘颜色对超出范围的区域进行着色
> - **REPEAT**则是平铺形式重复渲染
> - **MIRROR**则是在横向和纵向上以镜像的方式重复渲染位图。

2)ComposeShader(混合渲染)

ComposeShader(Shader shaderA, Shader shaderB, PorterDuff.Mode mode)

> 渲染效果的叠加，看到PorterDuff就知道什么了吧？比如将BitmapShader与LinearGradient的混合渲染 效果等。参数依次：
>
> - **shaderA**：第一种渲染效果
> - **shaderB**：第二种渲染效果
> - **mode**：两种渲染效果的叠加模式

3)LinearGradient(线性渲染)

LinearGradient(float x0, float y0, float x1, float y1, int[] colors, float[] positions, Shader.TileMode tile);

> 实现某一区域内颜色的线性渐变效果，参数依次是：
>
> - **x0**：渐变的起始点x坐标
> - **y0**：渐变的起始点y坐标
> - **x1**：渐变的终点x坐标
> - **y1**：渐变的终点y坐标
> - **colors**：渐变的颜色数组
> - **positions**：颜色数组的相对位置
> - **tile**：平铺方式

4)RadialGradient(环形渲染)

public RadialGradient (float x, float y, float radius, int[] colors, float[] positions, Shader.TileMode tile);

> 实现某一区域内颜色的环形渐变效果，参数依次是：
>
> - **x**：环形的圆心x坐标
> - **y**：环形的圆心y坐标
> - **radius**：环形的半径
> - **colors**：环形渐变的颜色数组
> - **positions**：指定颜色数组的相对位置
> - **tile**：平铺方式

5)SweepGradient(梯度渲染)

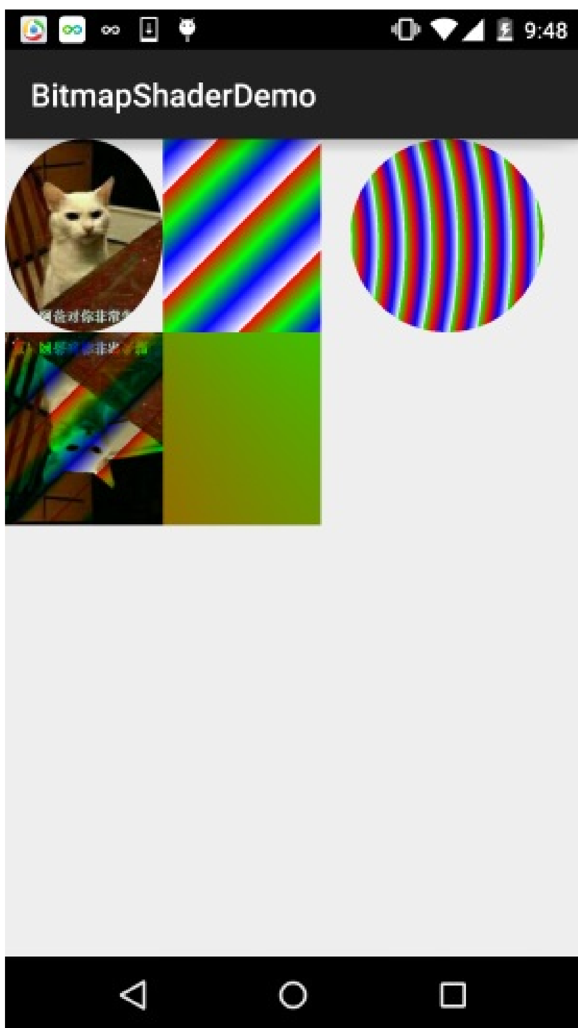
public SweepGradient (float cx, float cy, int[] colors, float[] positions)

> 扫描渲染，就是以某个点位中心旋转一周所形成的效果！参数依次是：
>
> - **cx**：扫描的中心x坐标
> - **cy**：扫描的中心y坐标
> - **colors**：梯度渐变的颜色数组
> - **positions**：指定颜色数组的相对位置

可能从文字上我们可以简单的知道下他们对应的一个大概作用，但是我们还是写个代码来 验证下他们所起的作用，毕竟有码(图)有真相吗~

2.使用代码示例：

运行效果图：



实现代码：

BitmapShaderView.java：

```
/**
 * Created by Jay on 2015/11/4 0030.
 */ public class BitmapShaderView extends View { private Bi
```

就那么一百来行代码，就不用解释了吧，如果觉得有疑惑的，动手试试~

3.本节代码下载：

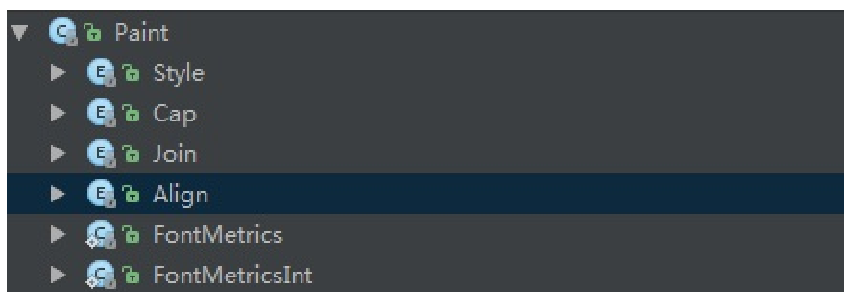
BitmapShaderDemo.zip

本节小结：

> 本节给大家介绍了Paint的另一个API：Shader(图像渲染)，又让我们的画笔增添了一种选择~ 如果你看到代码有疑惑，不懂把代码粘下，改改参数，就懂了~ 好的，本节就到这里，谢谢~

8.3.14 Paint几个枚举/常量值以及ShadowLayer阴影效果

本节引言：

> 在Android基础入门教程——8.3.1 三个绘图工具类详解Paint的方法参数那里我们就接触到 了这样几个东西：Paint.Style，Paint.Cap，Paint.Join等，这些都是Paint中的一些枚举值，相关 方法我们可以通过设置这些枚举值来设置特定效果比如：Style：画笔样式，Join图形结合方式等， 本节我们走进Paint的源码，我们来一一介绍这些枚举值，另外我们也顺道讲下这个ShadowLayer 设置带阴影效果的Paint！打开Paint类的源码，我们可以看到下述这些枚举值：
>
> 
>
>
> 好了，不BB，开始本节内容！

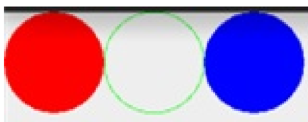
1.get枚举用法：

不知大家对枚举陌生还是熟悉，这里把贴下Paint.Style相关的调用代码(带有参构造方法的枚举)，让大家体会体会：

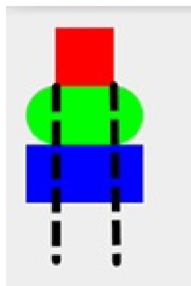
```
public enum Style {
    //定义枚举,通过括号赋值
    FILL            (0),
    STROKE          (1),
    FILL_AND_STROKE (2);
    //有参构造方法
    Style(int nativeInt) {
        this.nativeInt = nativeInt;
    }
    final int nativeInt;
}
//设置画笔Style的方法
public void setStyle(Style style) {
    native_setStyle(mNativePaint, style.nativeInt);
}
//JNI设置画笔风格的方法, 这里我们无需关注
private static native void native_setStyle(long native_object, int
```

下面我们一一来解释这些枚举值的作用！

1.Paint.Style

> 作用：画笔的样式 可选值：
>
> - **FILL**：填充内部(默认)
> - **STROKE**：只描边
> - **FILL_AND_STROKE**：填充内部与描边
>
> 方法调用：**setStyle(Paint.Style style)** 对应效果：
>
> 

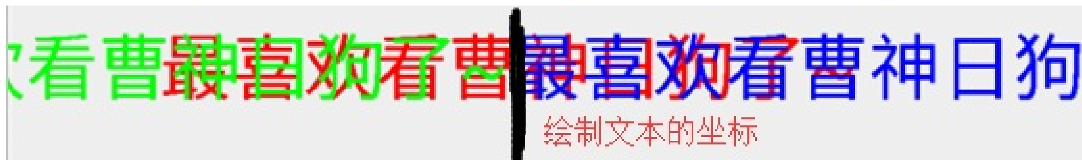
2.Paint.Cap

> 作用：笔触风格，设置画笔始末端的图形(画笔开始画的第一点与最后一点) 可选值：
>
> - **BUTT**：笔触是长方形且不超过路径(默认)
> - **ROUND**：笔触是圆形
> - **SQUARE**：笔触是正方形
>
> 方法调用：**setStrokeCap(Paint.Cap cap)**
>
> 对应效果：平时我们直接画的是第一个，其他两个会比普通的多一点而外的区域，第二个 是圆角，第三个是矩形！
>
> 

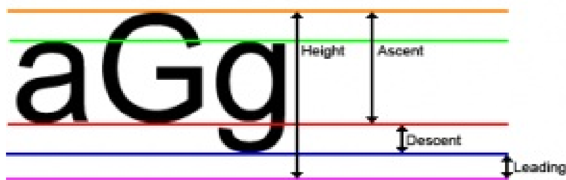
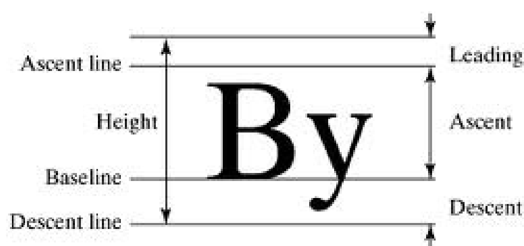
3.Paint.Join

> 作用：设置接合处的状态，比如你的线是由多条小线拼接而成，拼接处的形状 可选值：
>
> - **MITER**：接合处为锐角(默认)
> - **ROUND**：接合处为圆弧
> - **BEVEL**：接合处为直线
>
> 方法调用：**setStrokeJoin(Paint.Join join)**
>
> 一般圆弧用得多，可参见之前的擦掉美女衣服Demo的显示
>
> 另外还有个**setStrokeMiter(float miter)**是设置笔画的倾斜度，miter > = 0； 如：小时候用的铅笔，削的时候斜与垂直削出来的笔尖效果是不一样的。 主要是用来设置笔触的连接处的样式。可以和setStrokeJoin()来比较比较。

4.Paint.Align

> 作用：设置绘制文本的对其方式，就是相对于绘制文字的[x,y]起始坐标 可选值：
>
> - **LEFT**：在起始坐标的左边绘制文本
> - **RIGHT**：在起始坐标的右边绘制文本
> - **CENTER**：以其实坐标为中心绘制文本
>
> 方法调用：**setTextAlign(Paint.Align align)**
>
> 对应效果：另外可调用setTextSize()设置绘制文本的大小~
>
> 
>

5.Paint.FontMetrics和Paint.FontMetricsInt

> 字体属性及测量，另外这两个方法是一样的，只是后者取到的值是一个整形，这里我们选FontMetricsInt来给大家讲解下，有下面这五个常量值，这里参考的基准点是：下划线的位置(**Baseline**)
>
> - **top**：最高字符到baseline的距离，即ascent的最大值
> - **ascent**：字符最高处的距离到baseline的值
> - **descent**：下划线到字符最低处的距离
> - **bottom**：下划线到最低字符的距离，即descent的最大值
> - **leading**：上一行字符的descent到下一行的ascent之间的距离
>
> 我们看几个图帮助理解下：
>
> 
>
>
> 
>
>
> 
>
>
> 然后我们随意画一串字母，把这些值打印出来：**canvas.drawText("abcdefghijklnmopqrstuvwxyz", 400, 400, mPaint1); Log.e("HEHE", mPaint1.getFontMetricsInt().toString());** 运行下，我们可以看到，打印出来的Log如下：
>
> ```
> E/HEHE： FontMetricsInt: top=-34 ascent=-30 descent=8 bottom=9 leading=0
> ```
>
> 看完思考思考，画一画，应该不难理解！这里我们知道下就好，如果你想更 深入研究，可以参考下这篇：Android字符串进阶之三：字体属性及测量（FontMetrics）

6.ShadowLayer设置阴影效果

> 我们在TextView那一节就教过大家为TextView的文本设置阴影效果，而Paint其实也提供了设置 阴影效果的API：**setShadowLayer(float radius, float dx, float dy, int shadowColor)**
>
> 参数：radius为阴影的角度，dx和dy为阴影在x轴和y轴上的距离，shadowColor为阴影的颜色 我们可以写个非常简单的句子验证下：
>
> ```
> mPaint1.setShadowLayer(5,0,0,Color.BLACK);
> canvas.drawText("毕竟基神~", 400, 400, mPaint1);    //绘制文字
> ```

效果如下：

毕竟基神~

另外我们还可以调用**clearShadowLayer()**来清除这个阴影层~

本节小结：

> 好的，本节给大家讲解了下Paint里面的几个枚举值以及静态常量，以及ShadowLayer为画笔 设置阴影效果或调用clearShadowLayer()清除阴影层~其实这些东西都可以自己去看源码以及 文档，有疑惑就动手写个Demo，很多东西就自然一清二楚的了，嗯，就说这么多，谢谢~
>
> 另外，可能你不知道在哪看到了我的QQ，但是可以的话尽量加群好么，平时也要上班， 一个两个还好，一堆人，有心无力，有时帮忙解决问题，结果一天什么都没做，望各位体谅， 有问题加请加小猪群，群管理都是非常热心
>
> 的：**421858269~**

8.3.15 Paint API之——Typeface(字型)

> 本节带来Paint API系列的最后一个API，**Typeface**(字型)，由字义，我们大概可以猜到，这个 API是用来设置字体以及字体风格的，使用起来也非常的简单！下面我们来学习下Typeface的一些相关 的用法！
>
> 官方API文档：Typeface~
>
> 

1.字体的可选风格

> 四个整型常量：
>
> - **BOLD**：加粗
> - **ITALIC**：斜体
> - **BOLD_ITALIC**：粗斜体
> - **NORMAL**：正常

2.可选字体对象(Typeface)

> Android系统默认支持三种字体，分别为：**sans**，**serif**，**monospace** 而提供的可选静态对象值有五个：
>
> - **DEFAULT**：默认正常字体对象
> - **DEFAULT_BOLD**：默认的字体对象，注意:这实际上不可能是粗体的,这取决于字体设置。 由getStyle()来确定
> - **MONOSPACE**：monospace 字体风格
> - **SANS_SERIF**：sans serif字体风格
> - **SERIF**：serif字体风格

3.自定义创建字型

> 可能默认的三种字体并不能满足你，可能你喜欢MAC的字体——**Monaco**字体，你想让你APP 里的文字可以用这种字体，首先准备好我们的TTF文件，然后丢到**assets/font/**目录下 然后创建对应对象，关键代码如下：
>
> **Typeface typeFace =Typeface.createFromAsset(getAssets(),"font/MONACO.ttf");**

4.使用代码示例：

运行效果图：



自定义的View类：**MyView.java**：

```
/**
 * Created by Jay on 2015/11/5 0005.
 */
public class MyView extends View{

    private Paint mPaint1,mPaint2,mPaint3,mPaint4,mPaint5;
    private Context mContext;

    public MyView(Context context) {
        this(context,null);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        init();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
```

```
    }

    private void init(){
        mPaint1 = new Paint();
        mPaint2 = new Paint();
        mPaint3 = new Paint();
        mPaint4 = new Paint();
        mPaint5 = new Paint();

        mPaint1.setColor(Color.RED);
        mPaint2.setColor(Color.BLUE);
        mPaint3.setColor(Color.BLACK);
        mPaint4.setColor(Color.YELLOW);
        mPaint5.setColor(Color.GRAY);

        mPaint1.setTextSize(100);
        mPaint2.setTextSize(100);
        mPaint3.setTextSize(100);
        mPaint4.setTextSize(100);
        mPaint5.setTextSize(100);

        mPaint1.setTypeface(Typeface.DEFAULT_BOLD);
        mPaint2.setTypeface(Typeface.MONOSPACE);
        mPaint3.setTypeface(Typeface.SANS_SERIF);
        mPaint4.setTypeface(Typeface.SERIF);
        mPaint5.setTypeface(Typeface.createFromAsset(mContext.getA

    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawText("Coder-pig", 100, 100, mPaint1);
        canvas.drawText("Coder-pig", 100, 200, mPaint2);
        canvas.drawText("Coder-pig", 100, 300, mPaint3);
        canvas.drawText("Coder-pig", 100, 400, mPaint4);
        canvas.drawText("Coder-pig", 100, 500, mPaint5);
    }
}
```

恩呢，非常简单~就不解释了，要字体的可以自己百度或者下载示例代码~

本节示例代码下载：

TypefaceDemo.zip

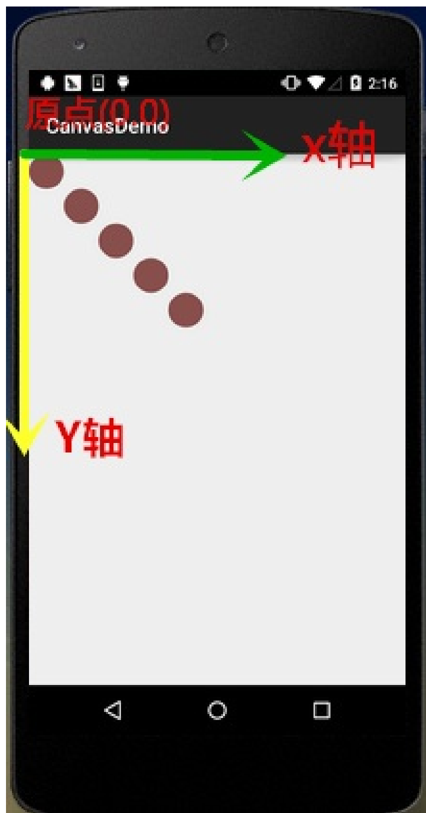
本节小结：

> 好的，一连十几节的Paint API详解就到这里了，应该已经涵盖大部分的可能会用到的API了，不知道你都Get了没，这些都是为我们进阶部分的自定义控件做铺垫~嗯，就说这么多，谢谢~

8.3.16 Canvas API详解(Part 1)

本节引言：

> 前面我们花了13小节详细地讲解了Android中Paint类大部分常用的API，本节开始我们来讲解 Canvas(画板)的一些常用API，我们在
>
> - 8.3.1 三个绘图工具类详解中已经列出了我们可供调用的一些方法，我们分下类：
> - **drawXxx**方法族：以一定的坐标值在当前画图区域画图，另外图层会叠加，即后面绘画的图层会覆盖前面绘画的图层。
> - **clipXXX**方法族：在当前的画图区域裁剪(clip)出一个新的画图区域，这个画图区域就是canvas对象的当前画图区域了。比如：clipRect(new Rect())，那么该矩形区域就是canvas的当前画图区域
> - **getXxx**方法族：获得与Canvas相关一些值，比如宽高，屏幕密度等。
> - **save**()，**restore**()，**saveLayer**()，**restoreToCount**()等保存恢复图层的方法
> - **translate**(平移)，**scale**(缩放)，**rotate**(旋转)，**skew**(倾斜)
>
> 当然还有其他一些零散的方法，嗯，从本节开始我会挑一些感觉有点意思的API来进行学习~
>
> 而本节先给大家带来的是**translate**(平移)，**scale**(缩放)，**rotate**(旋转)，**skew**(倾斜) 以及**save**()，**restore**()的详解！
>
> 官方API文档：Canvas
>
> 另外我们先要明确Canvas中X轴与Y轴的方向：



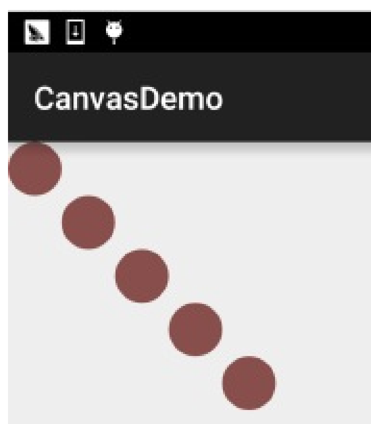
1.translate(平移)

> 方法：**translate**(float dx, float dy)
>
> 解析：平移，将画布的坐标原点向左右方向移动x，向上下方向移动y，canvas默认位置在(0,0)
>
> 参数：dx为水平方向的移动距离，dy为垂直方向的移动距离

使用示例：

```
for(int i=0; i <  5; i++)  { canvas.drawCircle(50,  50,  50, mPaint
```

运行效果：



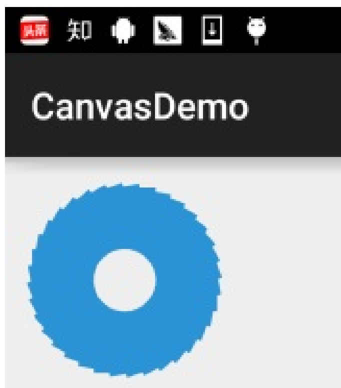
2.rotate(旋转)

> 方法：**rotate**(float degrees) / **rotate**(float degrees, float px, float py)
>
> 解析：围绕坐标原点旋转degrees度，值为正顺时针
>
> 参数：degrees为旋转角度，px和py为指定旋转的中心点坐标(px,py)

使用示例：

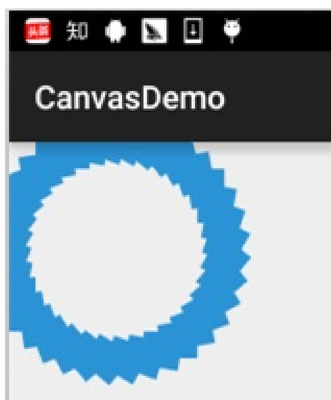
```
Rect rect =  new  Rect(50,0,150,50); canvas.translate(200,  200);
```

运行效果：



代码分析：

这里我们先调用了translate(200，200)将canvas的坐标原点移向了(200,200)，再进行绘制，所以我们 绘制的结果可以完整的在画布上显示出来，假如我们是为rotate设置了(10,200,200)，会是这样一个 结果：



有疑问是吧，这个涉及到Canvas多图层的概念，等等会讲~

3.scale(缩放)

> 方法：**scale**(float sx, float sy) / **scale**(float sx, float sy, float px, float py)
>
> 解析：对画布进行缩放
>
> 参数：sx为水平方向缩放比例，sy为竖直方向的缩放比例，px和py我也不知道，小数为缩小， 整数为放大

使用示例：

```
canvas.drawBitmap(bmp,0,0,mPaint); canvas.scale(0.8f,  0.8f); canva
```



运行效果：



4.skew(倾斜)

> 方法：**skew**(float sx, float sy)
>
> 解析：倾斜，也可以译作斜切，扭曲
>
> 参数：sx为x轴方向上倾斜的对应角度，sy为y轴方向上倾斜的对应角度，两个值都是tan值哦！ 都是tan值！都是tan值！比如要在x轴方向上倾斜60度，那么小数值对应:tan 60 = 根号3 = 1.732！

使用示例：

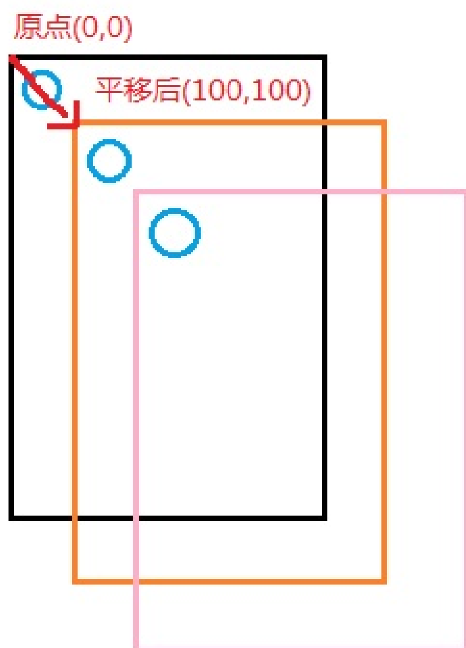
```
canvas.drawBitmap(bmp,0,0,mPaint); canvas.translate(200,  200); car
```

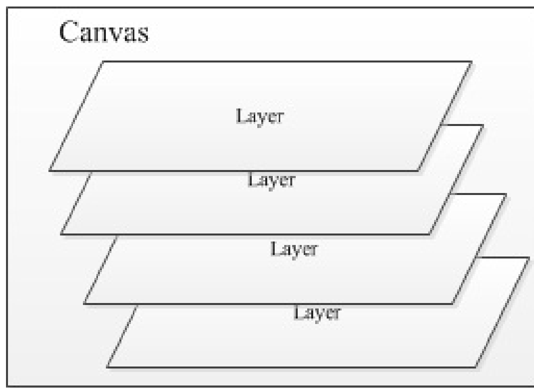


运行效果：



5.Canvas图层的概念以及save()和restore()详解

> 我们一般喜欢称呼Canvas为画布，童鞋们一直觉得Canvas就是一张简单的画纸，那么我想 问下多层的动画是怎么用canvas来完成的？上面那个translate平移的例子，为什么 drawCircle(50, 50, 50, mPaint); 参考坐标一直是(50,50)那为何会出现这样的效果？ 有疑惑的童鞋可能是一直将屏幕的概念与Canvas的概念混淆了，下面我们来还原下 调用translate的案发现场：
>
> 
>
>
> 如图，是画布坐标原点的每次分别在x，y轴上移动100；那么假如我们要重新回到(0,0) 点处绘制新的图形呢？怎么破，translate(-100,-100)的慢慢地平移回去？不会真的这么 纠结吧...
>
> 
>
> 好吧，不卖关子了，我们可以在做平移变换之前将当前canvas的状态进行保存，其实Canvas为 我们提供了图层(Layer)的支持，而这些Layer(图层)是按"栈结构"来进行管理的

> 
>
>
> 当我们调用**save()**方法，会保存当前Canvas的状态然后作为一个Layer(图层)，添加到Canvas栈中， 另外，这个Layer(图层)不是一个具体的类，就是一个概念性的东西而已！
>
> 而当我们调用**restore()**方法的时候，会恢复之前Canvas的状态，而此时Canvas的图层栈 会弹出栈顶的那个Layer，后继的Layer来到栈顶，此时的Canvas回复到此栈顶时保存的Canvas状态！
>
> 简单说就是：**save()**往栈压入一个**Layer**，**restore()**弹出栈顶的一个**Layer**，这个**Layer**代表**Canvas**的 状态！也就是说可以save()多次，也可以restore()多次，但是restore的调用次数不能大于save 否则会引发错误！这是网上大部分的说法，不过实际测试中并没有出现这样的问题，即使我restore的 次数多于save，也没有出现错误~目测是系统改了，等下测给大家看~  来来来，写个例子验证下save和restore的作用！

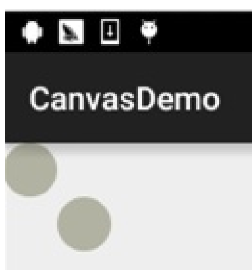
写个例子：

例子代码：

```
canvas.save();  //保存当前canvas的状态 canvas.translate(100,  100); c
```



运行结果：

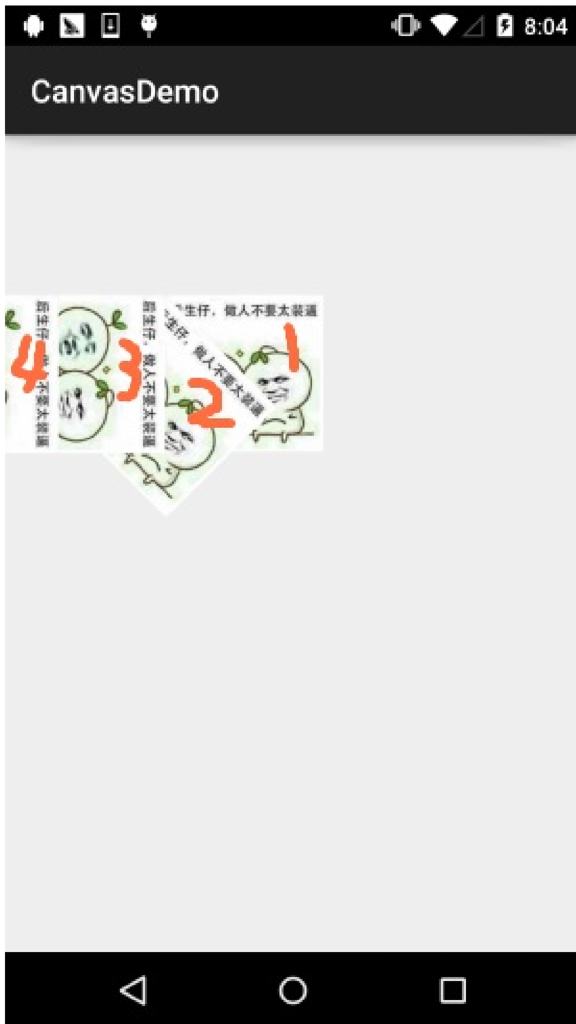


不用说什么了吧，代码和结果已经说明了一切，接着我们搞得复杂点，来一发 多个save()和restore()！

例子代码：

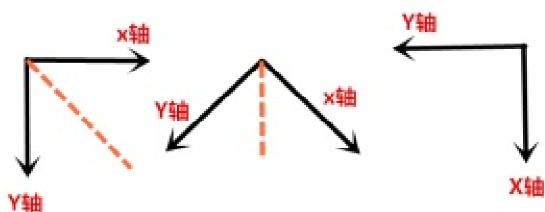
```
canvas.save(); canvas.translate(300,  300); canvas.drawBitmap(bmp,
```

运行结果：



结果分析：

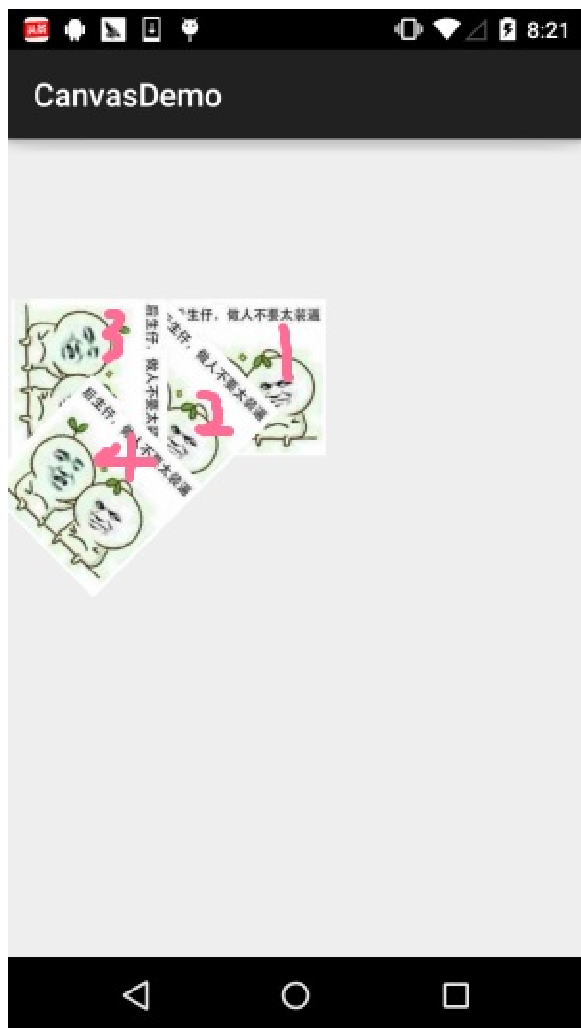
首先平移(300,300)画图，然后旋转45度画图，再接着旋转45度画图，接着平移(0,200)，期间每次画图前都save()一下，看到这里你可能有个疑问，最后这个平移不是y移动200 么，怎么变成向左了？嘿嘿，我会告诉你rotate()旋转的是整个坐标轴么？坐标轴的 变化：



嗯，rotate()弄懂了是吧，那就行，接着我们来试试restore咯~我们在最后绘图的前面 加两个restore()！

```
canvas.restore(); canvas.restore(); canvas.translate(0,  200); canv
```

运行结果：



不说什么，自己体会，再加多个**restore()**！



有点意思，再来，继续加**restore()**



嗯，好像不可以再写**restore**了是吧，因为我们只save了四次，按照网上的说法，这会报错的，真的是这样吗？这里我们调用Canvas给我们提供的一个获得当前栈中有多少个Layer的方法：**getSaveCount()**；然后在save()和restore()的前后都 加一个Log将栈中Layer的层数打印出来：

当前层数：2
当前层数：3
当前层数：4
当前层数：5
当前层数：4
当前层数：3
当前层数：2
当前层数：1
当前层数：1
当前层数：1
当前层数：1

结果真是喜闻乐见，毕竟实践出真知，可能是Canvas改过吧，或者其他原因，这里要看源码才知道了，时间关系，这里我们知道下restore的次数可以比save多就好了， 但是还是建议restore的次数还是少于save，以避免造成不必要的问题~ 至于进栈和出栈的流程我就不话了，笔者自己动笔画画，非常容易理解！

6.saveLayer()与restoreToCount()讲解

> 其实这两个方法和save以及restore大同小异，只是在后者的基础上多了一些东东而已，比如saveLayer()，有下面多个重载方法：
>
> | | |
> |---|---|
> | saveLayer (RectF bounds, Paint paint, int saveFlags) | This behaves the same as save(), but in addition it allocates and redirects drawing to an offscreen bitmap. |
> | saveLayer (RectF bounds, Paint paint) | Convenience for saveLayer(bounds, paint, ALL_SAVE_FLAG) |
> | saveLayer (float left, float top, float right, float bottom, Paint paint) | Convenience for saveLayer(left, top, right, bottom, paint, ALL_SAVE_FLAG) |
> | saveLayer (float left, float top, float right, float bottom, Paint paint, int saveFlags) | Helper version of saveLayer() that takes 4 values rather than a RectF. |
> | saveLayerAlpha (RectF bounds, int alpha, int saveFlags) | This behaves the same as save(), but in addition it allocates and redirects drawing to an offscreen bitmap. |
> | saveLayerAlpha (RectF bounds, int alpha) | Convenience for saveLayerAlpha(bounds, alpha, ALL_SAVE_FLAG) |
> | saveLayerAlpha (float left, float top, float right, float bottom, int alpha, int saveFlags) | Helper for saveLayerAlpha() that takes 4 values instead of a RectF. |
> | saveLayerAlpha (float left, float top, float right, float bottom, int alpha) | Helper for saveLayerAlpha(left, top, right, bottom, alpha, ALL_SAVE_FLAG) |
>
> 你可以理解为**save()**方法保存的是整个**Canvas**，而saveLayer()则可以选择性的保存某个区域的状态，另外，我们看到餐宿和中有个:**int saveFlags**，这个是设置改保存那个对象的！可选值有：

标记	说明
ALL_SAVE_FLAG	保存全部的状态
CLIP_SAVE_FLAG	保存裁剪的某个区域的状态
CLIP_TO_LAYER_SAVE_FLAG	保存预先设置的范围里的状态
FULL_COLOR_LAYER_SAVE_FLAG	保存彩色涂层
HAS_ALPHA_LAYER_SAVE_FLAG	不透明图层保存
MATRIX_SAVE_FLAG	Matrix信息(translate，rotate，scale，skew)的状态保存

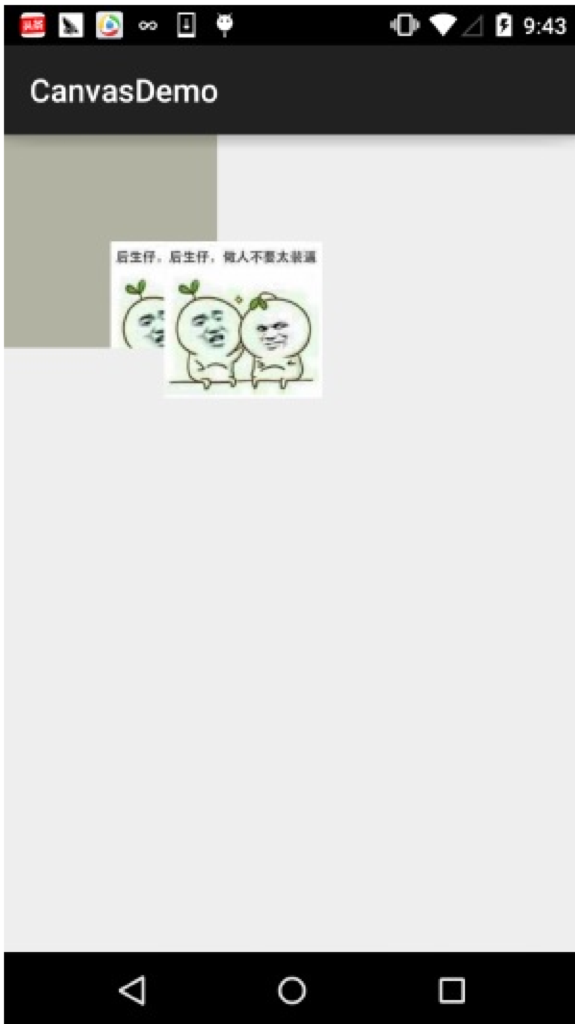
PS:上述说明有点问题，笔者英语水平低，可能说错，如果有知道的，请务必指正提出，谢谢~

这里我们写个例子来验证下：我们选用**CLIP_TO_LAYER_SAVE_FLAG**模式来写个例子

实现代码：

```
RectF bounds = new RectF(0, 0, 400, 400); canvas.saveLayer(bou
```

运行结果：



关于saveLayer()后面用到再详解研究吧~这里先知道个大概~

接着到这个**restoreToCount(int)**，这个更简单，直接传入要恢复到的Layer层数，直接就跳到对应的那一层，同时会将该层上面所有的Layer踢出栈，让该层 成为栈顶~！比起你写多个restore()方便快捷多了~

7.本节代码示例下载：

嗯，代码是写着测试的，要来也没多大意思，不过可能读者还是想要，就贴下链接吧！

代码下载：CanvasDemo.zip 可能你们要的是这个图吧！哈哈~



本节小结：

> 本节是纠结了几天才写出来的，因为笔者一开始对这个Canvas图层的概念也不是很清晰， 今天下午做完事捋了捋思路，晚上再加加班终于把这篇东西写出来了，相信应该能帮助 大家更清楚的理解Canvas，进阶自定义控件时也不会一头雾水~嘿嘿，本节就到这里， 如果有写错的地方欢迎提出，万分感谢~

参考文献： AndroidのCanvasを使いこなす！ – 基本的な描画

8.3.17 Canvas API详解(Part 2)剪切方法合集

本节引言：

> 本节继续带来Android绘图系列详解之Canvas API详解(Part 2)，今天要讲解的是Canvas 中的ClipXxx方法族！我们可以看到文档中给我们提供的Clip方法有三种类型：**clipPath**( )，**clipRect**( )，**clipRegion**( )；
>
> 通过Path，Rect，Region的不同组合，几乎可以支持任意形状的裁剪区域！
>
> **Path**：可以是开放或闭合的曲线，线构成的复杂的集合图形
>
> **Rect**：矩形区域
>
> **Region**：可以理解为区域组合，比如可以将两个区域相加，相减，并，疑惑等！
>
> Region.Op定义了Region支持的区域间运算种类！等下我们会讲到， 另外要说一点，我们平时理解的剪切可能是对已经存在的图形进行Clip，但是Android中对 Canvas进行Clip，是要在画图前进行的，如果画图后再对Canvas进行Clip的话将不会影响 到已经画好的图形，记住Clip是针对Canvas而非图形！ 嗯，不BB，直接开始本节内容！

官方**API**文档：Canvas

1.Region.Op组合方式详解

> 其实难点无非这个，Region代表着区域，表示的是Canvas图层上的某一块封闭区域！当然，有时间你可以自己慢慢去扣这个类，而我们一般关注的只是他的一个枚举值：**Op**
>
> ```
> // the native values for these must match up with the enum in SkRegion.h
> public enum Op {
>     DIFFERENCE(0),
>     INTERSECT(1),
>     UNION(2),
>     XOR(3),
>     REVERSE_DIFFERENCE(4),
>     REPLACE(5);
>
>     Op(int nativeInt) { this.nativeInt = nativeInt; }
>
>     /**
>      * @hide
>      */
>     public final int nativeInt;
> }
> ```
>
> 下面我们来看看个个枚举值所起的作用：我们假设两个裁剪区域A和B，那么我们调用Region.Op对应的枚举值：
>
> **DIFFERENCE**：A和B的差集范围，即A - B，只有在此范围内的绘制内容才会被显示；
>
> **INTERSECT**：即A和B的交集范围，只有在此范围内的绘制内容才会被显示
>
> **UNION**：即A和B的并集范围，即两者所包括的范围的绘制内容都会被显示；
>
> **XOR**：A和B的补集范围，此例中即A除去B以外的范围，只有在此范围内的绘制内容才会被显示；
>
> **REVERSE_DIFFERENCE**：B和A的差集范围，即B - A，只有在此范围内的绘制内容才会被显示；
>
> **REPLACE**：不论A和B的集合状况，B的范围将全部进行显示，如果和A有交集，则将覆盖A的交集范围；
>
> 如果你学过集合，那么画个Venn(韦恩图)就一清二楚了，没学过？没事，我们写个例子来试试 对应的结果~！写个初始化画笔以及画矩形的方法:

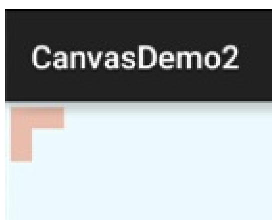
```
private void init() {
    mPaint = new Paint();
    mPaint.setAntiAlias(true);
    mPaint.setStrokeWidth(6);
    mPaint.setColor(getResources().getColor(R.color.blush));
}

private void drawScene(Canvas canvas){
    canvas.drawRect(0, 0, 200, 200, mPaint);
}
```

Op.DIFFERENCE：

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(50, 50, 150, 150, Region.Op.DIFFERENCE); //第二个
drawScene(canvas);
```

结果：



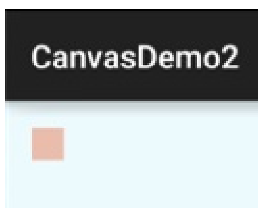
先后在(10,10)以及(50,50)为起点，裁剪了两个100*100的矩形，得出的裁剪结果是：

A和B的差集 = A - (A和B相交的部分)

Op.INTERSECT：

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(50, 50, 150, 150, Region.Op.INTERSECT); //第二个
drawScene(canvas);
```

结果：

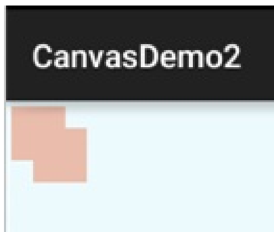


先后在(10,10)以及(50,50)为起点，裁剪了两个100100的矩形，得出的裁剪结果是： *A和B的交集 = A和B相交的部分

Op.UNION：

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(40, 40, 140, 140, Region.Op.UNION); //第二个
drawScene(canvas);
```

结果：

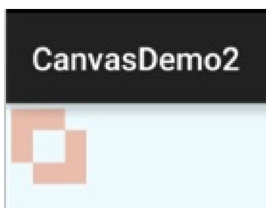


先后在(10,10)以及(50,50)为起点，裁剪了两个100100的矩形，得出的裁剪结果是： *A和B的并集 = A的区域 + B的区域

Op.XOR：

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(50, 50, 150, 150, Region.Op.XOR); //第二个
drawScene(canvas);
```

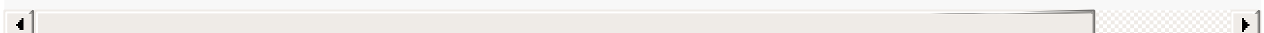
结果：



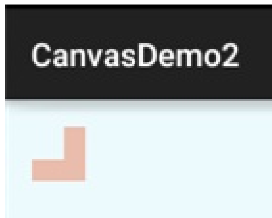
先后在(10,10)以及(50,50)为起点，裁剪了两个100100的矩形，得出的裁剪结果是： *A和B的补集 = A和B的合集 - A和B的交集

Op.REVERSE_DIFFERENCE：

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(50, 50, 150, 150, Region.Op.REVERSE_DIFFERENCE); //
drawScene(canvas);
```



结果：

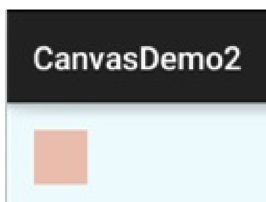


先后在(10,10)以及(50,50)为起点，裁剪了两个100*100*的矩形，得出的裁剪结果是： *B和A的差集 = B - A和B的交集

Op.REPLACE

```
canvas.clipRect(10, 10, 110, 110);        //第一个
canvas.clipRect(50, 50, 150, 150, Region.Op.REPLACE); //第二个
drawScene(canvas);
```

结果：

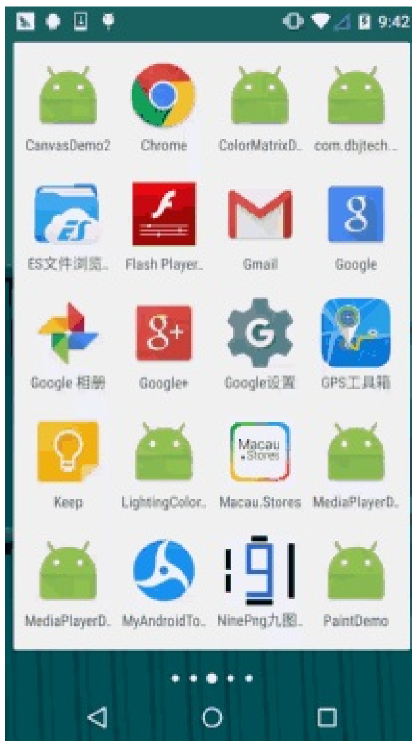


先后在(10,10)以及(50,50)为起点，裁剪了两个100*100*的矩形，得出的裁剪结果是： *不论A和B的集合状况，B的范围将全部进行显示，如果和A有交集，则将覆盖A的交集范围；

2.Region.Op使用实例：

例子参考自：Android 2D Graphics学习（二）、Canvas篇2、Canvas裁剪和Region、RegionIterator

运行效果图：



关键部分代码 **MyView.java**：

```
/**
 * Created by Jay on 2015/11/10 0010.
 */
public class MyView extends View{

    private Bitmap mBitmap = null;
    private int limitLength = 0;     //
    private int width;
    private int heigth;
    private static final int CLIP_HEIGHT = 50;

    private boolean status = HIDE;//显示还是隐藏的状态，最开始为HIDE
    private static final boolean SHOW = true;//显示图片
    private static final boolean HIDE = false;//隐藏图片

    public MyView(Context context) {
        this(context, null);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
        limitLength = width = mBitmap.getWidth();
        heigth = mBitmap.getHeight();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {
        Region region = new Region();
        int i = 0;
        while (i * CLIP_HEIGHT <= heigth) {//计算clip的区域
            if (i % 2 == 0) {
                region.union(new Rect(0, i * CLIP_HEIGHT, limitLeng
            } else {
                region.union(new Rect(width - limitLength, i * CLIP
                        * CLIP_HEIGHT));
            }
            i++;
        }
        canvas.clipRegion(region);
        canvas.drawBitmap(mBitmap, 0, 0, new Paint());
        if (status == HIDE) {//如果此时是隐藏
            limitLength -= 10;
            if(limitLength <= 0)
                status=SHOW;
        } else {//如果此时是显示
            limitLength += 5;
            if(limitLength >= width)
                status=HIDE;
        }
        invalidate();
    }
}
```

实现分析：

> 初始化的时候获得宽高，然后循环，可以理解把图片分割成一条条的线，循环条件是：i * 每条的高度 不大于高度，然后线又分两种情况，调用的是Region的union，其实就是结合方式为UNINO的剪切方式 而已，最后是对此时图片的是否显示做下判断，隐藏和显示的情况做不同的处理，最后调用invalidate() 重绘！还是蛮简单的，自己理解理解吧~

另外要说一点：Canvas的变换对clipRegion没有作用

3.clipRect方法详解：

clipRect提供了七个重载方法：

clipRect(Rect rect, Region.Op op) Modify the current clip with the specified rectangle, which is express
clipRect(RectF rect, Region.Op op) Modify the current clip with the specified rectangle.
clipRect(int left, int top, int right, int bottom) Intersect the current clip with the specified rectangle, which is expre
clipRect(float left, float top, float right, float bottom) Intersect the current clip with the specified rectangle, which is expre
clipRect(RectF rect) Intersect the current clip with the specified rectangle, which is expre
clipRect(float left, float top, float right, float bottom, Region.Op op) Modify the current clip with the specified rectangle, which is express
clipRect(Rect rect) Intersect the current clip with the specified rectangle, which is expre

参数介绍如下：

> **rect**：Rect对象，用于定义裁剪区的范围，Rect和RectF功能类似，精度和提供的方法不同而已
>
> **left**：矩形裁剪区的左边位置
>
> **top**：矩形裁剪区的上边位置
>
> **right**：矩形裁剪区的右边位置
>
> **bottom**：矩形裁剪区的下边位置
>
> **op**：裁剪区域的组合方式
>
> 上述四个值可以是浮点型或者整型

使用示例：

```
mPaint = new Paint();
mPaint.setAntiAlias(true);
mPaint.setColor(Color.BLACK);
mPaint.setTextSize(60);

canvas.translate(300,300);
canvas.clipRect(100, 100, 300, 300);                //设置显示范围
canvas.drawColor(Color.WHITE);                      //白色背景
canvas.drawText("双11, 继续吃我的狗粮...", 150, 300, mPaint); //绘制字
```

运行结果：



从上面的例子，不知道你发现了没？ clipRect会受Canvas变换的影响，白色区域是不花的区域，所以clipRect裁剪的是画布， 而我们的绘制是在这个裁剪后的画布上进行的！超过该区域的不显示！

4.clipPath方法详解：

相比起clipRect，clipPath就只有两个重载方法，使用方法非常简单，自己绘制一个Paht然后传入即可！

clipPath(Path path)	Intersect the current clip with the specified path.
clipPath(Path path, Region.Op op)	Modify the current clip with the specified path.

使用示例：

这里复用我们以前在ImageView那里写的圆形ImageView的例子~

实现代码：

自定义ImageView：RoundImageView.java

```
/**
 * Created by coder-pig on 2015/7/18 0018.
 */
public class RoundImageView extends ImageView {

    private Bitmap mBitmap;
    private Rect mRect = new Rect();
    private PaintFlagsDrawFilter pdf = new PaintFlagsDrawFilter(0,
    private Paint mPaint = new Paint();
    private Path mPath=new Path();
    public RoundImageView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    //传入一个Bitmap对象
    public void setBitmap(Bitmap bitmap) {
        this.mBitmap = bitmap;
    }

    private void init() {
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setFlags(Paint.ANTI_ALIAS_FLAG);
        mPaint.setAntiAlias(true);// 抗锯尺
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if(mBitmap == null)
        {
            return;
        }
        mRect.set(0,0,getWidth(),getHeight());
        canvas.save();
        canvas.setDrawFilter(pdf);
        mPath.addCircle(getWidth() / 2, getWidth() / 2, getHeight()
        canvas.clipPath(mPath, Region.Op.REPLACE);
        canvas.drawBitmap(mBitmap, null, mRect, mPaint);
        canvas.restore();
    }
}
```

布局代码：**activity_main.xml**:

```
<com.jay.demo.imageviewdemo.RoundImageView
        android:id="@+id/img_round"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_margin="5px"/>
```

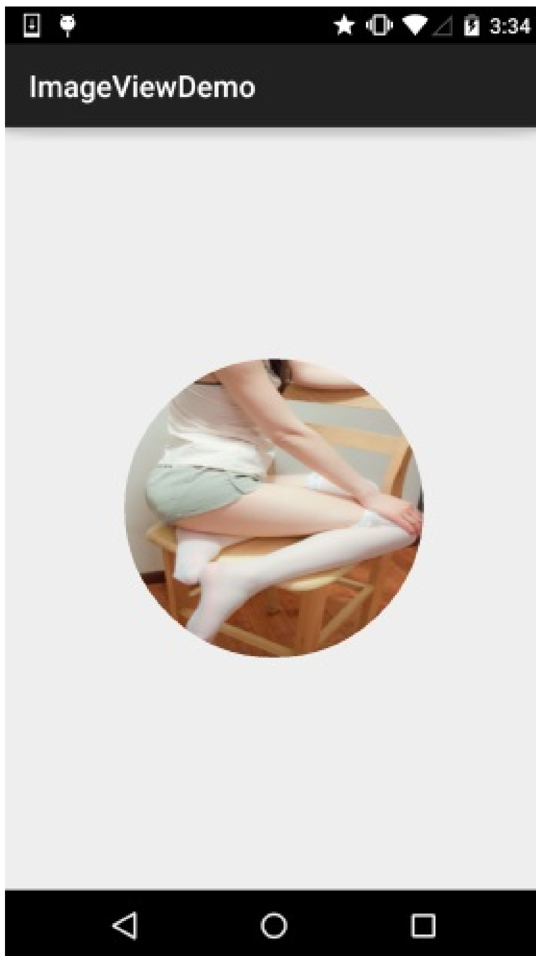
MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    private RoundImageView img_round;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        img_round = (RoundImageView) findViewById(R.id.img_round);
        Bitmap bitmap = BitmapFactory.decodeResource(getResources()
        img_round.setBitmap(bitmap);
    }
}
```

运行效果图：



另外使用该方法制作的圆角ImageView会有锯齿明显，即使你为Paint，Canvas设置了 抗锯齿也没用~假如你要求高的，可以使用Xfermode-PorterDuff设置图像混排来实现， 基本没锯齿，可见：Android基础入门教程——8.3.6 Paint API之——Xfermode与PorterDuff详解(三)

5.本节示例代码下载：

CanvasDemo2.zip

XfermodeDemo1.zip

本节小结：

> 好的，本节给大家讲解了下Canvas中剪切有个的三个方法：clipPath( )，clipRect( )， clipRegion( )，难点应该是在最后一个上，六种不同的Op组合方式，其实也不难，集合 的概念而已，放在开头，消化了就好，而clipPath( )，clipRect( )则没什么难点~ 对喔，今天双11，不知道你剁手了没~

8.3.18 Canvas API详解(Part 3)Matrix和drawBitmapMash

本节引言：

> 在Canvas的API文档中，我们看到这样一个方法：**drawBitmap**(Bitmap bitmap, **Matrix** matrix, Paint paint)
>
> 这个Matrix可是有大文章的，前面我们在学Paint的API中的ColorFilter中曾讲过ColorMatrix 颜色矩阵，一个4 * 5 的矩阵，我们可以通过修改矩阵值来修改色调，饱和度等！ 而今天讲的这个Matrix可以结合其他API来控制图形，组件的变换。比如Canvas就提供了上面的 这个drawBitmap来实现矩阵变换的效果！下面我们来慢慢研究这个东东~
>
> 官方API文档：Matrix

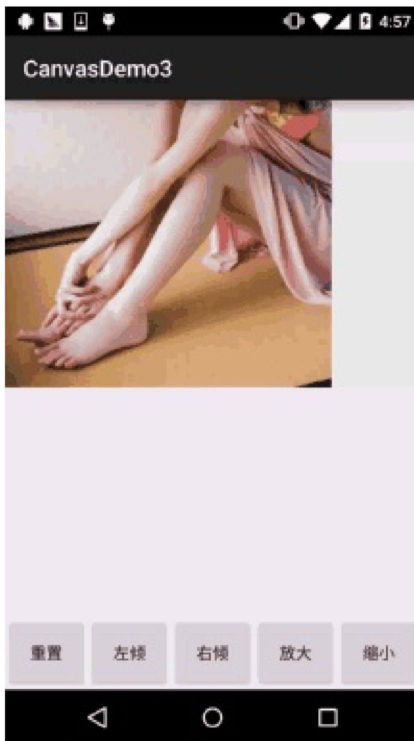
1.Matrix中的几个常用的变换方法

> - **setTranslate**(float dx, float dy)：控制Matrix进行平移
> - **setRotate**(float degrees, float px, float py)：旋转，参数依次是:旋转角度，轴心(x,y)
> - **setScale**(float sx, float sy, float px, float py):缩放， 参数依次是：X，Y轴上的缩放比例；缩放的轴心
> - **setSkew**(float kx, float ky)：倾斜(扭曲)，参数依次是：X，Y轴上的缩放比例

其实和Canvas变换的方法基本一致，在为Matrix设置了上面的变换后，调用Canvas的 drawBitmap()方法调用矩阵就好~

2.Matrix使用示例：

运行效果图：



代码实现：

MyView.java：

```
/**
 * Created by Jay on 2015/11/11 0011.
 */
public class MyView extends View {

    private Bitmap mBitmap;
    private Matrix matrix = new Matrix();
    private float sx = 0.0f;          //设置倾斜度
    private int width,height;         //位图宽高
    private float scale = 1.0f;       //缩放比例
    private int method = 0;

    public MyView(Context context) {
        this(context, null);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
    }

    private void init() {
        mBitmap = BitmapFactory.decodeResource(getResources(), R.m
```

```
        width = mBitmap.getWidth();
        height = mBitmap.getHeight();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        switch (method){
            case 0:
                matrix.reset();
                break;
            case 1:
                sx += 0.1;
                matrix.setSkew(sx,0);
                break;
            case 2:
                sx -= 0.1;
                matrix.setSkew(sx,0);
                break;
            case 3:
                if(scale < 2.0){
                    scale += 0.1;
                }
                matrix.setScale(scale,scale);
                break;
            case 4:
                if(scale > 0.5){
                    scale -= 0.1;
                }
                matrix.setScale(scale,scale);
                break;
        }
        //根据原始位图与Matrix创建新图片
        Bitmap bitmap = Bitmap.createBitmap(mBitmap,0,0,width,heigh
        canvas.drawBitmap(bitmap,matrix,null);    //绘制新位图
    }

    public void setMethod(int i){
        method = i;
        postInvalidate();
    }
}
```

布局代码:**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
```

```
        android:id="@+id/ly_bar"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:layout_alignParentBottom="true">

        <Button
            android:id="@+id/btn_reset"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="重置" />

        <Button
            android:id="@+id/btn_left"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="左倾" />

        <Button
            android:id="@+id/btn_right"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="右倾" />

        <Button
            android:id="@+id/btn_zoomin"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="放大" />

        <Button
            android:id="@+id/btn_zoomout"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:text="缩小" />
    </LinearLayout>

    <com.jay.canvasdemo3.MyView
        android:id="@+id/myView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/ly_bar" />

</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_reset;
    private Button btn_left;
    private Button btn_right;
    private Button btn_zoomin;
    private Button btn_zoomout;
    private MyView myView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_reset = (Button) findViewById(R.id.btn_reset);
        btn_left = (Button) findViewById(R.id.btn_left);
        btn_right = (Button) findViewById(R.id.btn_right);
        btn_zoomin = (Button) findViewById(R.id.btn_zoomin);
        btn_zoomout = (Button) findViewById(R.id.btn_zoomout);
        myView = (MyView) findViewById(R.id.myView);

        btn_reset.setOnClickListener(this);
        btn_left.setOnClickListener(this);
        btn_right.setOnClickListener(this);
        btn_zoomin.setOnClickListener(this);
        btn_zoomout.setOnClickListener(this);

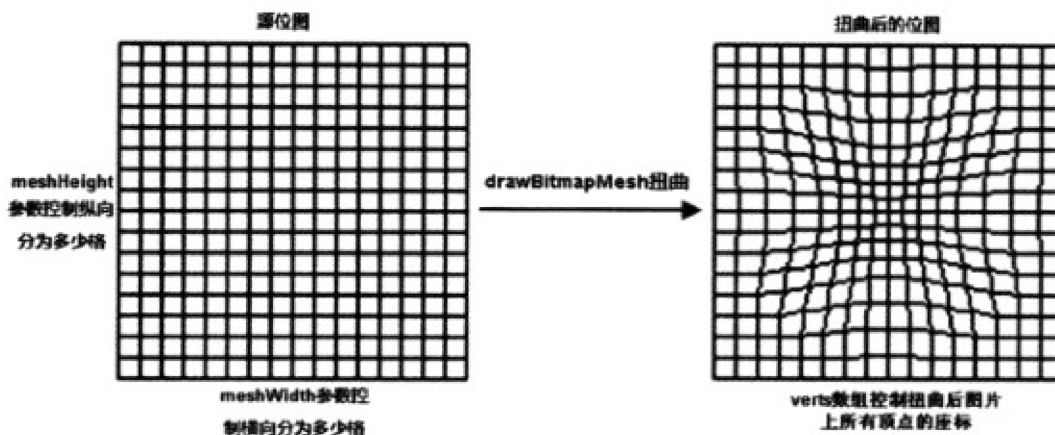
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_reset:
                myView.setMethod(0);
                break;
            case R.id.btn_left:
                myView.setMethod(1);
                break;
            case R.id.btn_right:
                myView.setMethod(2);
                break;
            case R.id.btn_zoomin:
                myView.setMethod(3);
                break;
            case R.id.btn_zoomout:
                myView.setMethod(4);
                break;
        }
    }
}
```

用法非常简单，就不解释了~

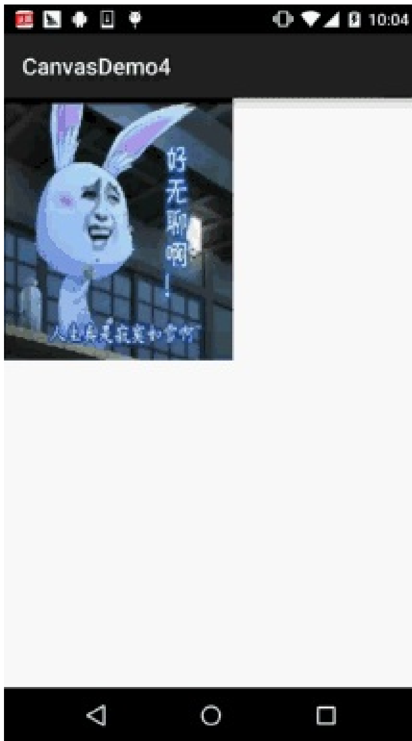
3.drawBitmapMash扭曲图像

> 在API文档中还有这样一个方法： **drawBitmapMesh**(Bitmap bitmap, int meshWidth, int meshHeight, float[] verts, int vertOffset, int[] colors, int colorOffset, Paint paint)
>
> 参数依次是：
>
> **bitmap**：需要扭曲的原位图
>
> **meshWidth/meshHeight**：在横/纵向上把原位图划分为多少格
>
> **verts**：长度为(meshWidth+1)*(meshHeight+2)的数组，他记录了扭曲后的位图各顶点(网格线交点) 位置，虽然他是一个一维数组，但是实际上它记录的数据是形如(x0,y0)，(x1,y1)..(xN,Yn)格式的数据， 这些数组元素控制对bitmap位图的扭曲效果
>
> **vertOffset**：控制verts数组从第几个数组元素开始对bitmap进行扭曲(忽略verOffset之前数据 的扭曲效果)



代码示例：

运行效果图：



代码实现：

```
/**
 * Created by Jay on 2015/11/11 0011.
 */
public class MyView extends View {

    //将水平和竖直方向上都划分为20格
    private final int WIDTH = 20;
    private final int HEIGHT = 20;
    private final int COUNT = (WIDTH + 1) * (HEIGHT + 1);  //记录该
    private final float[] verts = new float[COUNT * 2];    //扭曲前2
    private final float[] orig = new float[COUNT * 2];    //扭曲后21
    private Bitmap mBitmap;
    private float bH,bW;

    public MyView(Context context) {
        this(context, null);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public MyView(Context context, AttributeSet attrs, int defStyle
        super(context, attrs, defStyleAttr);
    }

    private void init() {
        mBitmap = BitmapFactory.decodeResource(getResources(), R.mi
```

```
        bH = mBitmap.getWidth();
        bW = mBitmap.getHeight();
        int index = 0;
        //初始化orig和verts数组。
        for (int y = 0; y <= HEIGHT; y++)
        {
            float fy = bH * y / HEIGHT;
            for (int x = 0; x <= WIDTH; x++)
            {
                float fx = bW * x / WIDTH;
                orig[index * 2 + 0] = verts[index * 2 + 0] = fx;
                orig[index * 2 + 1] = verts[index * 2 + 1] = fy;
                index += 1;
            }
        }
        //设置背景色
        setBackgroundColor(Color.WHITE);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmapMesh(mBitmap, WIDTH, HEIGHT, verts
                , 0, null, 0, null);
    }

    //工具方法，用于根据触摸事件的位置计算verts数组里各元素的值
    private void warp(float cx, float cy)
    {
        for (int i = 0; i < COUNT * 2; i += 2)
        {
            float dx = cx - orig[i + 0];
            float dy = cy - orig[i + 1];
            float dd = dx * dx + dy * dy;
            //计算每个座标点与当前点（cx、cy）之间的距离
            float d = (float)Math.sqrt(dd);
            //计算扭曲度，距离当前点（cx、cy）越远，扭曲度越小
            float pull = 80000 / ((float) (dd * d));
            //对verts数组（保存bitmap上21 * 21个点经过扭曲后的座标）重新赋
            if (pull >= 1)
            {
                verts[i + 0] = cx;
                verts[i + 1] = cy;
            }
            else
            {
                //控制各顶点向触摸事件发生点偏移
                verts[i + 0] = orig[i + 0] + dx * pull;
                verts[i + 1] = orig[i + 1] + dy * pull;
            }
        }
        //通知View组件重绘
        invalidate();
    }
```

```
    @Override
    public boolean onTouchEvent(MotionEvent event)
    {
        //调用warp方法根据触摸屏事件的座标点来扭曲verts数组
        warp(event.getX(), event.getY());
        return true;
    }

}
```

实现流程分析：

> 首先你要弄清楚，这个verts数组存储的是什么？比如 verts[0]和verts1，这两个相邻的元素其实表示的就是我们第一个点的x坐标和y坐标！知道这一点，你就知道为什么有21 *21*个点，以及为什么数组长度等于这个值 2！初始化部分也就懂了！
>
> 接着我们再来看看根据触摸事件计算verts数组元素的值的实现：获得触摸点的x,y坐标，拿这个值去减对应点的x,y只，计算出触摸点和每个坐标点的距离 然后计算所谓的扭曲度：80000 / ((float) (dd * d))；扭曲度 >= 1的，直接让该坐标 点指向这个触摸点，< 1的，则让各个顶点向触摸点发生偏移，然后再调用invalidate()重绘~ 大概就这样~多思考思考，如果还是不理解就算了~知道有这东西就好！

4.本节示例下载：

CanvasDemo3.zip

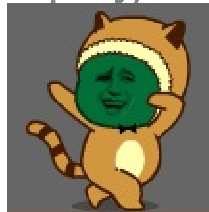
CanvasDemo4.zip

本节小结：

> 本节内容大部分摘自——李刚《Android》疯狂讲义，可能稍微容易理解一点吧~ Matrix应该大部分的童鞋都能看懂，而drawBitmapMash扭曲图像则可能需要一点 时间消化消化，看不懂也没什么哈~嗯，本节就到这里，谢谢
>
> 

8.4.1 Android动画合集之帧动画

本节引言：

> 从本节开始我们来探究Android中的动画，毕竟在APP中添加上一些动画，会让我们的应用变得 很炫，比如最简单的关开Activity，当然自定义控件动画肯定必不可少啦~而Android中的动画 分为三大类，逐帧动画(**Frame**)以及补间动画(**Tween**)，还有Android 3.0以后引入的属性动画 (**Property**)，而本节给大家带来的是第一种动画——逐帧动画的一个基本使用~ 

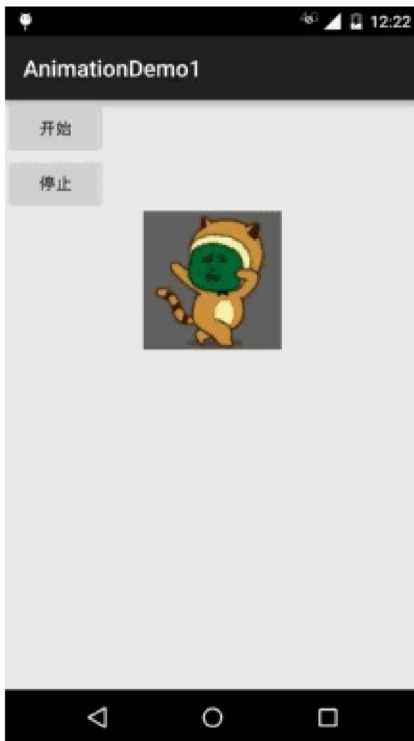
1.帧动画概念以及用法

> 帧动画非常容易理解，其实就是简单的由N张静态图片收集起来，然后我们通过控制依次显示 这些图片，因为人眼"视觉残留"的原因，会让我们造成动画的"错觉"，跟放电影的原理一样！
>
> 而Android中实现帧动画，一般我们会用到前面讲解到的一个Drawable：AnimationDrawable 先编写好Drawable，然后代码中调用start()以及stop()开始或停止播放动画~
>
> 当然我们也可以在Java代码中创建逐帧动画，创建AnimationDrawable对象，然后调用 addFrame(Drawable frame,int duration)向动画中添加帧，接着调用start()和stop()而已~
>
> 下面我们来写两个例子体会下帧动画的效果以及熟悉下用法

2.使用示例：

示例一：最简单的例子：

运行效果图：



代码实现：

首先编写我们的动画文件，非常简单，先在res下创建一个anim目录，接着开始撸我们的 动画文件：**miao_gif.xml**： 这里的android:oneshot是设置动画是否只是播放一次，true只播放一次，false循环播放！

```
<?xml version="1.0" encoding="utf-8"?> <animation-list xmlns:andr
```

动画文件有了，接着到我们的布局文件：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/an
```

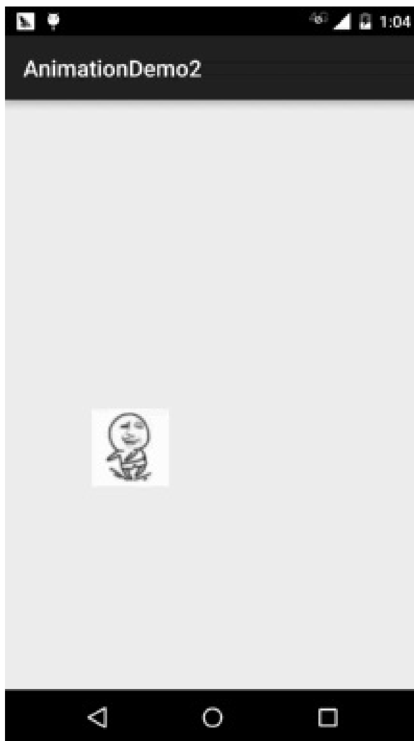
最后是我们的**MainActivity.java**，这里在这里控制动画的开始以及暂停：

```
public class MainActivity extends AppCompatActivity implements
```

好的，非常的简单哈~

示例二：在指定地方播放帧动画

运行效果图：



代码实现：

依旧是先上我们的动画文件:**anim_zhuan.xml**：

```
<animation-list  xmlns:android="http://schemas.android.com/apk/res/
```

接着我们来写一个自定义的ImageView：**FrameView.java**，这里通过反射获得当前播放的帧，然后是否为最后一帧，是的话隐藏控件！

```
/**
 * Created by Jay on 2015/11/15 0015.
 */  public  class  FrameView  extends  ImageView  {  private  Anin
```

最后是我们的**MainActivity.java**，创建一个FrameLayout，添加View，对触摸事件中按下的 事件做处理，显示控件以及开启动画~

```
public  class  MainActivity  extends  AppCompatActivity  {  private
```

好的，同样很简单哈~

3.本节示例代码和Gif帧提取工具下载

AnimationDemo1.zip

AnimationDemo2.zip

Gif帧提取工具

本节小结：

> 好的，本节给大家介绍了一下Android三种动画中最简单的帧动画，实际开发用的并不多 不过学多点东西，以后也多个思路嘛~好的，本节就到这里，谢谢~

8.4.2 Android动画合集之补间动画

本节引言：

> 本节带来的是Android三种动画中的第二种——补间动画(Tween)，和前面学的帧动画不同，帧动画 是通过连续播放图片来模拟动画效果，而补间动画开发者只需指定动画开始，以及动画结束"关键帧"， 而动画变化的"中间帧"则由系统计算并补齐！好了，开始本节学习~

1.补间动画的分类和Interpolator

Andoird所支持的补间动画效果有如下这五种，或者说四种吧，第五种是前面几种的组合而已~

> - **AlphaAnimation**：透明度渐变效果，创建时许指定开始以及结束透明度，还有动画的持续 时间，透明度的变化范围(0,1)，0是完全透明，1是完全不透明；对应**<alpha/>**标签！
> - **ScaleAnimation**：缩放渐变效果，创建时需指定开始以及结束的缩放比，以及缩放参考点， 还有动画的持续时间；对应**<scale/>**标签！
> - **TranslateAnimation**：位移渐变效果，创建时指定起始以及结束位置，并指定动画的持续 时间即可；对应**<translate/>**标签！
> - **RotateAnimation**：旋转渐变效果，创建时指定动画起始以及结束的旋转角度，以及动画 持续时间和旋转的轴心；对应**<rotate/>**标签
> - **AnimationSet**：组合渐变，就是前面多种渐变的组合，对应**<set/>**标签

在开始讲解各种动画的用法之前，我们先要来讲解一个东西：**Interpolator**

> 用来控制动画的变化速度，可以理解成动画渲染器，当然我们也可以自己实现Interpolator 接口，自行来控制动画的变化速度，而Android中已经为我们提供了五个可供选择的实现类：
>
> - **LinearInterpolator**：动画以均匀的速度改变
> - **AccelerateInterpolator**：在动画开始的地方改变速度较慢，然后开始加速
> - **AccelerateDecelerateInterpolator**：在动画开始、结束的地方改变速度较慢，中间时加速
> - **CycleInterpolator**：动画循环播放特定次数，变化速度按正弦曲线改变：Math.sin(2 *mCycles* Math.PI * input)
> - **DecelerateInterpolator**：在动画开始的地方改变速度较快，然后开始减速
> - **AnticipateInterpolator**：反向，先向相反方向改变一段再加速播放
> - **AnticipateOvershootInterpolator**：开始的时候向后然后向前甩一定值后返回最后的值
> - **BounceInterpolator**： 跳跃，快到目的值时值会跳跃，如目的值100，后面的值可能依次为85，77，70，80，90，100
> - **OvershottInterpolator**：回弹，最后超出目的值然后缓慢改变到目的值
>
> 而这个东东，我们一般是在写动画xml文件时会用到，属性是：**android:interpolator**， 而上面对应的值是：**@android:anim/linear_interpolator**，其实就是驼峰命名法变下划线而已AccelerateDecelerateInterpolator对应：@android:anim/accelerate_decelerate_interpolator！

2.各种动画的详细讲解

这里的**android:duration**都是动画的持续时间，单位是毫秒~

1）AlphaAnimation(透明度渐变)

anim_alpha.xml：

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_decelerate_inte
    android:fromAlpha="1.0"
    android:toAlpha="0.1"
    android:duration="2000"/>
```

属性解释：

fromAlpha :起始透明度 **toAlpha**:结束透明度 透明度的范围为：0-1，完全透明-完全不透明

2）ScaleAnimation(缩放渐变)

anim_scale.xml：

```
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:fromXScale="0.2"
    android:toXScale="1.5"
    android:fromYScale="0.2"
    android:toYScale="1.5"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000"/>
```

属性解释：

> - **fromXScale/fromYScale**：沿着X轴/Y轴缩放的起始比例
> - **toXScale/toYScale**：沿着X轴/Y轴缩放的结束比例
> - **pivotX/pivotY**：缩放的中轴点X/Y坐标，即距离自身左边缘的位置，比如50%就是以图像的 中心为中轴点

3）TranslateAnimation(位移渐变)

anim_translate.xml：

```
<translate xmlns:android="http://schemas.android.com/apk/res/androi
    android:interpolator="@android:anim/accelerate_decelerate_inte
    android:fromXDelta="0"
    android:toXDelta="320"
    android:fromYDelta="0"
    android:toYDelta="0"
    android:duration="2000"/>
```

属性解释：

> - **fromXDelta/fromYDelta**：动画起始位置的X/Y坐标
> - **toXDelta/toYDelta**：动画结束位置的X/Y坐标

4）RotateAnimation(旋转渐变)

anim_rotate.xml：

```
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_decelerate_inter
    android:fromDegrees="0"
    android:toDegrees="360"
    android:duration="1000"
    android:repeatCount="1"
    android:repeatMode="reverse"/>
```

属性解释：

> - **fromDegrees/toDegrees**：旋转的起始/结束角度
> - **repeatCount**：旋转的次数，默认值为0，代表一次，假如是其他值，比如3，则旋转4次 另外，值为-1或者infinite时，表示动画永不停止
> - **repeatMode**：设置重复模式，默认**restart**，但只有当repeatCount大于0或者infinite或-1时 才有效。还可以设置成**reverse**，表示偶数次显示动画时会做方向相反的运动！

5）AnimationSet(组合渐变)

非常简单，就是前面几个动画组合到一起而已~

anim_set.xml：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/decelerate_interpolator"
    android:shareInterpolator="true" >

    <scale
        android:duration="2000"
        android:fromXScale="0.2"
        android:fromYScale="0.2"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="1.5"
        android:toYScale="1.5" />

    <rotate
        android:duration="1000"
        android:fromDegrees="0"
        android:repeatCount="1"
        android:repeatMode="reverse"
        android:toDegrees="360" />

    <translate
        android:duration="2000"
        android:fromXDelta="0"
        android:fromYDelta="0"
        android:toXDelta="320"
        android:toYDelta="0" />

    <alpha
        android:duration="2000"
        android:fromAlpha="1.0"
        android:toAlpha="0.1" />

</set>
```

3.写个例子来体验下

好的，下面我们就用上面写的动画来写一个例子，让我们体会体会何为补间动画：首先来个简单的布局：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_alpha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="透明度渐变" />

    <Button
        android:id="@+id/btn_scale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="缩放渐变" />

    <Button
        android:id="@+id/btn_tran"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="位移渐变" />

    <Button
        android:id="@+id/btn_rotate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="旋转渐变" />

    <Button
        android:id="@+id/btn_set"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="组合渐变" />

    <ImageView
        android:id="@+id/img_show"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="48dp"
        android:src="@mipmap/img_face" />

</LinearLayout>
```

好哒，接着到我们的**MainActivity.java**，同样非常简单，只需调用 AnimationUtils.loadAnimation() 加载动画，然后我们的View控件调用 startAnimation开启动画即可~

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_alpha;
    private Button btn_scale;
    private Button btn_tran;
    private Button btn_rotate;
    private Button btn_set;
    private ImageView img_show;
    private Animation animation = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_alpha = (Button) findViewById(R.id.btn_alpha);
        btn_scale = (Button) findViewById(R.id.btn_scale);
        btn_tran = (Button) findViewById(R.id.btn_tran);
        btn_rotate = (Button) findViewById(R.id.btn_rotate);
        btn_set = (Button) findViewById(R.id.btn_set);
        img_show = (ImageView) findViewById(R.id.img_show);

        btn_alpha.setOnClickListener(this);
        btn_scale.setOnClickListener(this);
        btn_tran.setOnClickListener(this);
        btn_rotate.setOnClickListener(this);
        btn_set.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_alpha:
                animation = AnimationUtils.loadAnimation(this,
                        R.anim.anim_alpha);
                img_show.startAnimation(animation);
                break;
            case R.id.btn_scale:
                animation = AnimationUtils.loadAnimation(this,
                        R.anim.anim_scale);
                img_show.startAnimation(animation);
                break;
            case R.id.btn_tran:
                animation = AnimationUtils.loadAnimation(this,
                        R.anim.anim_translate);
                img_show.startAnimation(animation);
                break;
            case R.id.btn_rotate:
```

```
                animation = AnimationUtils.loadAnimation(this,
                        R.anim.anim_rotate);
                img_show.startAnimation(animation);
                break;
            case R.id.btn_set:
                animation = AnimationUtils.loadAnimation(this,
                        R.anim.anim_set);
                img_show.startAnimation(animation);
                break;
        }
    }
}
```

运行效果图：



嘿嘿，有点意思是吧，还不动手试试，改点东西，或者自由组合动画，做出酷炫的效果吧~

4.动画状态的监听

> 我们可以对动画的执行状态进行监听，调用动画对象的：

- **setAnimationListener(new AnimationListener())**方法，重写下面的三个方法：
- **onAnimationStart**()：动画开始
- **onAnimtaionRepeat**()：动画重复
- **onAnimationEnd**()：动画结束

即可完成动画执行状态的监听~

5.为View动态设置动画效果

> 先调用**AnimationUtils.loadAnimation**(动画xml文件)，然后View控件调用startAnimation(anim) 开始动画~这是静态加载的方式，当然你也可以直接创建一个动画对象，用Java代码完成设置，再调用 startAnimation开启动画~

6.为Fragment设置过渡动画

这里要注意一点，就是Fragment是使用的**v4**包还是**app**包下的Fragment！ 我们可以调用**FragmentTransaction**对象的**setTransition(int transit)** 为Fragment指定标准的过场动画，transit的可选值如下：

- **TRANSIT_NONE**：无动画
- **TRANSIT_FRAGMENT_OPEN**：打开形式的动画
- **TRANSIT_FRAGMENT_CLOSE**：关闭形式的动画

上面的标准过程动画是两个都可以调用的，而不同的地方则在于自定义转场动画

setCustomAnimations()方法！

- **app**包下的**Fragment**： **setCustomAnimations(int enter, int exit, int popEnter, int popExit)** 分别是添加，移除，入栈，以及出栈时的动画！ 另外要注意一点的是，对应的动画类型是：属性动画(Property)，就是动画文件 的根标签要是：<**objectAnimator**>，<**valueAnimator**>或者是前面两者放到一个<**set**>里；

- **v4**包下的**Fragment**： v4包下的则支持两种setCustomAnimations()

```
public abstract FragmentTransaction setCustomAnimations(@AnimRes int enter,
        @AnimRes int exit);

/**
 * Set specific animation resources to run for the fragments that are
 * entering and exiting in this transaction. The <code>popEnter</code>
 * and <code>popExit</code> animations will be played for enter/exit
 * operations specifically when popping the back stack.
 */
public abstract FragmentTransaction setCustomAnimations(@AnimRes int enter,
        @AnimRes int exit, @AnimRes int popEnter, @AnimRes int popExit);
```

另外要注意一点的是，对应的动画类型是：补间动画(Tween)，和上面的View一样~

可能你会有疑惑，你怎么知道对应的动画类型，其实只要你到Fragment源码那里找下：

onCreateAnimation()方法的一个返回值就知道了：

v4包：

```
public Animation onCreateAnimation(int transit, boolean enter, int nextAnim)
```

app包：

```
public Animator onCreateAnimator(int transit, boolean enter, int nextAnim)
```

7.为Activity设置过场动画

Activty设置过场动画非常简单，调用的方法是：**overridePendingTransition**(int enterAnim, int exitAnim)

用法很简单：在**startActivity(intent)**或者**finish()**后添加

参数依次是：新**Activity**进场时的动画，以及旧**Activity**退场时的动画

下面提供几种比较简单而且常用的过场动画供大家使用~

```
对应效果:
淡入淡出效果
放大淡出效果
转动淡出效果1
转动淡出效果2
左上角展开淡出效果
压缩变小淡出效果
右往左推出效果
下往上推出效果
左右交错效果
放大淡出效果
缩小效果
上下交错效果
```

> 下载传送门：Activity常用过渡动画.zip

8.写个进入APP后登陆注册按钮从底部弹出动画效果的例子：

运行效果图：



代码实现：

首先是我们的布局文件：**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DDE2E3"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/start_ctrl"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="vertical"
        android:visibility="gone">

        <Button
            android:id="@+id/start_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#F26968"
            android:gravity="center"
            android:paddingBottom="15dp"
            android:paddingTop="15dp"
            android:text="登陆"
            android:textColor="#FFFFFF"
            android:textSize="18sp" />

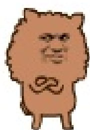
        <Button
            android:id="@+id/start_register"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#323339"
            android:gravity="center"
            android:paddingBottom="15dp"
            android:paddingTop="15dp"
            android:text="注册"
            android:textColor="#FFFFFF"
            android:textSize="18sp" />
    </LinearLayout>

</RelativeLayout>
```

接着是**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity {
    private LinearLayout start_ctrl;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        start_ctrl = (LinearLayout) findViewById(R.id.start_ctrl);
        //设置动画，从自身位置的最下端向上滑动了自身的高度，持续时间为500ms
        final TranslateAnimation ctrlAnimation = new TranslateAnima
                TranslateAnimation.RELATIVE_TO_SELF, 0, TranslateAn
                TranslateAnimation.RELATIVE_TO_SELF, 1, TranslateAn
        ctrlAnimation.setDuration(500l);     //设置动画的过渡时间
        start_ctrl.postDelayed(new Runnable() {
            @Override
            public void run() {
                start_ctrl.setVisibility(View.VISIBLE);
                start_ctrl.startAnimation(ctrlAnimation);
            }
        }, 2000);
    }
}
```

注释写得很清楚了，这里就不BB解释了，如果你对TranslateAnimation.RELATIVE_TO_SELF这个有疑惑，请自己谷歌或者百度，限于篇幅(我懒)，这里就不写了，蛮简单的~



9.本节代码示例下载

AnimationDemo3.zip

AnimationDemo4.zip

本节小结：

> 本节给大家细细地讲解了下Android中的第二种动画(渐变动画)，四种动画的详解，以及 设置动画监听器，还有如何为View，Fragment和Activity设置动画，最后还写了一个进入后 从APP底部弹出登陆按钮和注册按钮的例子，篇幅可能有点长，不过都非常容易理解，相信 大家看完都能够收获满满~!好的，本节就到这里，谢谢~

8.4.3 Android动画合集之属性动画-初见

本节引言：

本节给带来的是Android动画中的第三种动画——属性动画(Property Animation)，记得在上一节8.4.2 Android动画合集之补间动画为Fragment 设置过渡动画的时候，说过，App包和V4包下的Fragment调用setCustomAnimations()对应的 动画类型是不一样的，v4包下的是**Animation**，而app包下的是**Animator**；

Animation一般动画就是我们前面学的帧动画和补间动画！**Animator**则是本节要讲的属性动画！

关于属性动画，大牛郭大叔已经写了三篇非常好的总结文，写得非常赞，就没必要重复造轮子了，不过这里还是过一遍，大部分内容参考的下面三篇文章：

Android属性动画完全解析(上)，初识属性动画的基本用法

Android属性动画完全解析(中)，ValueAnimator和ObjectAnimator的高级用法

Android属性动画完全解析(下)，Interpolator和ViewPropertyAnimator的用法

写的非常好，或者说你可以直接跳过本文去看上面的三篇文章~

当然，你愿意看我叨叨逼的话，也很欢迎，好了，开始本节内容吧~

1.属性动画概念叨叨逼

不BB，直接上图，就是这么暴力~

Android属性动画(Property animation)解析

1.为什么要引入属性动画？

1.补间动画功能比较单调，只有四种动画(透明度，旋转，倾斜和位移)
2.补间动画针对的对象只是UI控件
3.补间动画只是改变View的显示效果，不会去改变View的属性！

eg:左边的按钮移到右边，但是此时的按钮其实还停留在左边，
假如你去点右面的按钮，是不会触发按钮的点击事件的~

2.属性动画又是怎样一种东西？

1.Andoid 3.0引入，可以说是补间动画的增强版，不止可以实现四种动画效果，可以定义任何属性的变化；
2.执行动画的对象不只是UI控件，可以对任何对象执行动画(不管是否显示在屏幕上)
3.属性动画通过对目标对象进行赋值来修改其属性，上面那个按钮问题就不存在了~

3.相关API

Animator

创建属性动画的基类，一般不会直接用，一般用他的两个子类！

ValueAnimator

上面也说了，属性动画是通过不断地修改值来实现的，而初始值和结束值间的过度动画就是由该类来负责计算的。内部采用一种时间循环的机制来计算值与值之间的动画过度，我们只需将初始值以及结束值提供给该类，并告诉它动画所需时长，该类就会自动帮我们完成从初始值平滑过渡到结束值这样的效果！除此之外，该类还负责管理动画播放次数，播放模式，以及对动画设置监听器等！

ObjectAnimator

ValueAnimator的子类，允许我们对指定对象的属性执行动画，用起来更加简单，实际中用得较多。当然某些场合下，可能还是需要用到ValueAnimator

AnimatorSet

Animator的子类，用于组合多个Animator，并制定多个Animator按照次序播放，还是同时播放

Evaluator(计算器)

告诉动画系统如何从初始值过度到结束值，提供了下述几种Evaluator：
- IntEvaluator：用于计算int类型属性值的计算器
- FloatEvaluator：用于计算float类型属性值的计算器
- ArgbEvaluator：用于计算十六进制形式表示的颜色值的计算器
- TypeEvaluator：计算器的接口，我们可以实现该接口来完成自定义计算器

2.ValueAnimator简单使用

使用流程：

- 1.调用ValueAnimator的**ofInt**()，**ofFloat**()或**ofObject**()静态方法创建ValueAnimator实例
- 2.调用实例的setXxx方法设置动画持续时间，插值方式，重复次数等
- 3.调用实例的**addUpdateListener**添加**AnimatorUpdateListener**监听器，在该监听器中 可以获得ValueAnimator计算出来的值，你可以值应用到指定对象上~
- 4.调用实例的**start()**方法开启动画！ 另外我们可以看到ofInt和ofFloat都有个这样的参数：float/int... values代表可以多个值！

使用示例：



代码实现：

布局文件：**activity_main.xml**，非常简单，四个按钮，一个ImageView

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:id="@+id/ly_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_one"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="动画1" />

    <Button
        android:id="@+id/btn_two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="动画2" />

    <Button
        android:id="@+id/btn_three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="动画3" />

    <Button
        android:id="@+id/btn_four"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="动画4" />

    <ImageView
        android:id="@+id/img_babi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@mipmap/img_babi" />

</LinearLayout>
```

接着到**MainActivity.java**， 首先需要一个修改View位置的方法，这里调用**moveView**()设置左边和上边的起始坐标以及宽高！

接着定义了四个动画，分别是：直线移动，缩放，旋转加透明，以及圆形旋转！

然后通过按钮触发对应的动画~

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_one;
    private Button btn_two;
```

```
    private Button btn_three;
    private Button btn_four;
    private LinearLayout ly_root;
    private ImageView img_babi;
    private int width;
    private int height;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        ly_root = (LinearLayout) findViewById(R.id.ly_root);
        btn_one = (Button) findViewById(R.id.btn_one);
        btn_two = (Button) findViewById(R.id.btn_two);
        btn_three = (Button) findViewById(R.id.btn_three);
        btn_four = (Button) findViewById(R.id.btn_four);
        img_babi = (ImageView) findViewById(R.id.img_babi);

        btn_one.setOnClickListener(this);
        btn_two.setOnClickListener(this);
        btn_three.setOnClickListener(this);
        btn_four.setOnClickListener(this);
        img_babi.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_one:
                lineAnimator();
                break;
            case R.id.btn_two:
                scaleAnimator();
                break;
            case R.id.btn_three:
                raAnimator();
                break;
            case R.id.btn_four:
                circleAnimator();
                break;
            case R.id.img_babi:
                Toast.makeText(MainActivity.this, "不愧是coder-pig~'
                break;
        }
    }

    //定义一个修改ImageView位置的方法
    private void moveView(View view, int rawX, int rawY) {
        int left = rawX - img_babi.getWidth() / 2;
```

```
        int top = rawY - img_babi.getHeight();
        int width = left + view.getWidth();
        int height = top + view.getHeight();
        view.layout(left, top, width, height);
    }

    //定义属性动画的方法：

    //按轨迹方程来运动
    private void lineAnimator() {
        width = ly_root.getWidth();
        height = ly_root.getHeight();
        ValueAnimator xValue = ValueAnimator.ofInt(height,0,height
        xValue.setDuration(3000L);
        xValue.addUpdateListener(new ValueAnimator.AnimatorUpdateLi
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                // 轨迹方程 x = width / 2
                int y = (Integer) animation.getAnimatedValue();
                int x = width / 2;
                moveView(img_babi, x, y);
            }
        });
        xValue.setInterpolator(new LinearInterpolator());
        xValue.start();
    }

    //缩放效果
    private void scaleAnimator(){

        //这里故意用两个是想让大家体会下组合动画怎么用而已~
        final float scale = 0.5f;
        AnimatorSet scaleSet = new AnimatorSet();
        ValueAnimator valueAnimatorSmall = ValueAnimator.ofFloat(1.
        valueAnimatorSmall.setDuration(500);

        ValueAnimator valueAnimatorLarge = ValueAnimator.ofFloat(sc
        valueAnimatorLarge.setDuration(500);

        valueAnimatorSmall.addUpdateListener(new ValueAnimator.Anim
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                float scale = (Float) animation.getAnimatedValue();
                img_babi.setScaleX(scale);
                img_babi.setScaleY(scale);
            }
        });
        valueAnimatorLarge.addUpdateListener(new ValueAnimator.Anim
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                float scale = (Float) animation.getAnimatedValue();
                img_babi.setScaleX(scale);
                img_babi.setScaleY(scale);
```

```
            }
        });

        scaleSet.play(valueAnimatorLarge).after(valueAnimatorSmall)
        scaleSet.start();

        //其实可以一个就搞定的
//        ValueAnimator vValue = ValueAnimator.ofFloat(1.0f, 0.6f,
//        vValue.setDuration(1000L);
//        vValue.addUpdateListener(new ValueAnimator.AnimatorUpdate
//            @Override
//            public void onAnimationUpdate(ValueAnimator animation
//                float scale = (Float) animation.getAnimatedValue(
//                img_babi.setScaleX(scale);
//                img_babi.setScaleY(scale);
//            }
//        });
//        vValue.setInterpolator(new LinearInterpolator());
//        vValue.start();
    }

    //旋转的同时透明度变化
    private void raAnimator(){
        ValueAnimator rValue = ValueAnimator.ofInt(0, 360);
        rValue.setDuration(1000L);
        rValue.addUpdateListener(new ValueAnimator.AnimatorUpdateLi
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                int rotateValue = (Integer) animation.getAnimatedVa
                img_babi.setRotation(rotateValue);
                float fractionValue = animation.getAnimatedFraction
                img_babi.setAlpha(fractionValue);
            }
        });
        rValue.setInterpolator(new DecelerateInterpolator());
        rValue.start();
    }

    //圆形旋转
    protected void circleAnimator() {
        width = ly_root.getWidth();
        height = ly_root.getHeight();
        final int R = width / 4;
        ValueAnimator tValue = ValueAnimator.ofFloat(0,
                (float) (2.0f * Math.PI));
        tValue.setDuration(1000);
        tValue.addUpdateListener(new ValueAnimator.AnimatorUpdateLi
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                // 圆的参数方程 x = R * sin(t) y = R * cos(t)
                float t = (Float) animation.getAnimatedValue();
                int x = (int) (R * Math.sin(t) + width / 2);
                int y = (int) (R * Math.cos(t) + height / 2);
```

```
                moveView(img_babi, x, y);
            }
        });
        tValue.setInterpolator(new DecelerateInterpolator());
        tValue.start();
    }
}
```

好的，使用的流程非常简单，先创建ValueAnimator对象，调用ValueAnimator.ofInt/ofFloat 获得，然后设置动画持续时间，**addUpdateListener**添加**AnimatorUpdateListener**事件监听， 然后使用参数**animation**的**getAnimatedValue**()获得当前的值，然后我们可以拿着这个值 来修改View的一些属性，从而形成所谓的动画效果，接着设置setInterpolator动画渲染模式， 最后调用start()开始动画的播放~

卧槽，直线方程，圆的参数方程，我都开始方了，这不是高数的东西么， 挂科学渣连三角函数都忘了...

例子参考自github：MoveViewValueAnimator

3.ObjectAnimator简单使用

比起ValueAnimator，ObjectAnimator显得更为易用，通过该类我们可以直接 对任意对象的任意属性进行动画操作！没错，是任意对象，而不单单只是View对象， 不断地对对象中的某个属性值进行赋值，然后根据对象属性值的改变再来决定如何展现 出来！比如为TextView设置如下动画： **ObjectAnimator.ofFloat(textview, "alpha", 1f, 0f);** 这里就是不断改变alpha的值，从1f - 0f，然后对象根据属性值的变化来刷新界面显示，从而 展现出淡入淡出的效果，而在TextView类中并没有alpha这个属性，ObjectAnimator内部机制是： 寻找传输的属性名对应的**get**和**set**方法~，而非找这个属性值！ 不信的话你可以到TextView的源码里找找是否有alpha这个属性！ 好的，下面我们利用ObjectAnimator来实现四种补间动画的效果吧~

运行效果图：



代码实现：

布局直接用的上面那个布局，加了个按钮，把ImageView换成了TextView，这里就不贴代码了， 直接上**MainActivity.java**部分的代码，其实都是大同小异的~

```
public class MainActivity extends AppCompatActivity implements View
    private Button btn_one;
    private Button btn_two;
    private Button btn_three;
    private Button btn_four;
    private Button btn_five;
    private LinearLayout ly_root;
    private TextView tv_pig;
    private int height;
    private ObjectAnimator animator1;
    private ObjectAnimator animator2;
    private ObjectAnimator animator3;
    private ObjectAnimator animator4;
    private AnimatorSet animSet;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
        initAnimator();
    }

    private void bindViews() {
        ly_root = (LinearLayout) findViewById(R.id.ly_root);
        btn_one = (Button) findViewById(R.id.btn_one);
```

```
        btn_two = (Button) findViewById(R.id.btn_two);
        btn_three = (Button) findViewById(R.id.btn_three);
        btn_four = (Button) findViewById(R.id.btn_four);
        btn_five = (Button) findViewById(R.id.btn_five);
        tv_pig = (TextView) findViewById(R.id.tv_pig);

        height = ly_root.getHeight();
        btn_one.setOnClickListener(this);
        btn_two.setOnClickListener(this);
        btn_three.setOnClickListener(this);
        btn_four.setOnClickListener(this);
        btn_five.setOnClickListener(this);
        tv_pig.setOnClickListener(this);
    }

    //初始化动画
    private void initAnimator() {
        animator1 = ObjectAnimator.ofFloat(tv_pig, "alpha", 1f, 0f,
        animator2 = ObjectAnimator.ofFloat(tv_pig, "rotation", 0f,
        animator3 = ObjectAnimator.ofFloat(tv_pig, "scaleX", 2f, 4f
        animator4 = ObjectAnimator.ofFloat(tv_pig, "translationY",
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_one:
                animator1.setDuration(3000l);
                animator1.start();
                break;
            case R.id.btn_two:
                animator2.setDuration(3000l);
                animator2.start();
                break;
            case R.id.btn_three:
                animator3.setDuration(3000l);
                animator3.start();
                break;
            case R.id.btn_four:
                animator4.setDuration(3000l);
                animator4.start();
                break;
            case R.id.btn_five:
                //将前面的动画集合到一起~
                animSet = new AnimatorSet();
                animSet.play(animator4).with(animator3).with(animat
                animSet.setDuration(5000l);
                animSet.start();
                break;
            case R.id.tv_pig:
                Toast.makeText(MainActivity.this, "不愧是coder-pig~"
                break;
        }
```

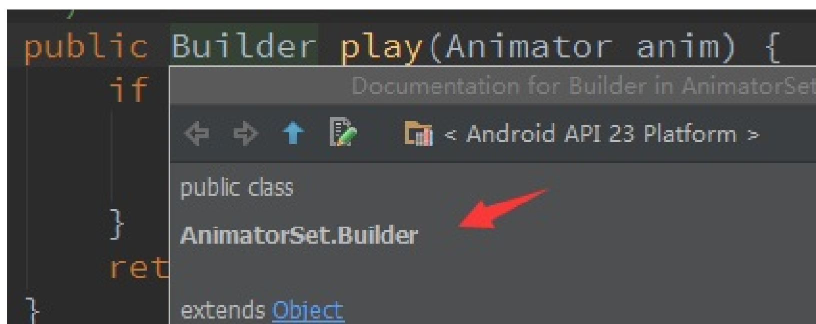
```
    }
}
```

用法也非常简单，上面涉及到的组合动画我们下面讲~

4.组合动画与AnimatorListener

从上面两个例子中我们都体验了一把组合动画，用到了**AnimatorSet**这个类！

我们调用的play()方法，然后传入第一个开始执行的动画，此时他会返回一个Builder类给我们：



接下来我们可以调用Builder给我们提供的四个方法，来组合其他的动画：

- **after**(Animator anim) 将现有动画插入到传入的动画之后执行
- **after**(long delay) 将现有动画延迟指定毫秒后执行
- **before**(Animator anim) 将现有动画插入到传入的动画之前执行
- **with**(Animator anim) 将现有动画和传入的动画同时执行

嗯，很简单，接下来要说下动画事件的监听，上面我们ValueAnimator的监听器是**AnimatorUpdateListener**，当值状态发生改变时候会回调**onAnimationUpdate**方法！

除了这种事件外还有：动画进行状态的监听~ **AnimatorListener**，我们可以调用**addListener**方法 添加监听器，然后重写下面四个回调方法：

- **onAnimationStart()**：动画开始
- **onAnimationRepeat()**：动画重复执行
- **onAnimationEnd()**：动画结束
- **onAnimationCancel()**：动画取消

没错，加入你真的用AnimatorListener的话，四个方法你都要重写，当然和前面的手势那一节一样， Android已经给我们提供好一个适配器类：**AnimatorListenerAdapter**，该类中已经把每个接口 方法都实现好了，所以我们这里只写一个回调方法也可以额！

5.使用XML来编写动画

使用XML来编写动画，画的时间可能比Java代码长一点，但是重用起来就轻松很多！ 对应的XML标签分别为：**<animator><objectAnimator><set>** 相关的属性解释如下：

- **android:ordering**：指定动画的播放顺序：sequentially(顺序执行)，together(同时执行)
- **android:duration**：动画的持续时间
- **android:propertyName**="x"：这里的x，还记得上面的"alpha"吗？加载动画的那个对象里需要 定义getx和setx的方法，objectAnimator就是通过这里来修改对象里的值的！
- **android:valueFrom**="1" ：动画起始的初始值
- **android:valueTo**="0" ：动画结束的最终值
- **android:valueType**="floatType"：变化值的数据类型

使用例子如下：

①从**0**到**100**平滑过渡的动画：

```
<animator xmlns:android="http://schemas.android.com/apk/res/androic
    android:valueFrom="0"
    android:valueTo="100"
    android:valueType="intType"/>
```

②将一个视图的**alpha**属性从**1**变成**0**：

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/a
    android:valueFrom="1"
    android:valueTo="0"
    android:valueType="floatType"
    android:propertyName="alpha"/>
```

③**set**动画使用演示：

```
<set android:ordering="sequentially" >
    <set>
        <objectAnimator
            android:duration="500"
            android:propertyName="x"
            android:valueTo="400"
            android:valueType="intType" />
        <objectAnimator
            android:duration="500"
            android:propertyName="y"
            android:valueTo="300"
            android:valueType="intType" />
    </set>
    <objectAnimator
        android:duration="500"
        android:propertyName="alpha"
        android:valueTo="1f" />
</set>
```

加载我们的动画文件：

```
AnimatorSet set = (AnimatorSet)AnimatorInflater.loadAnimator(mConte
             R.animator.property_animator);
animator.setTarget(view);
animator.start();
```

6.本节示例代码下载：

AnimatorDemo1.zip

AnimatorDemo2.zip

本节小结：

> 好的，本节给大家捋了一捋安卓中属性动画的基本用法，不知道你get了没，内容还是比较简单的，而且例子比较有趣，相信大家会喜欢，嗯，就说这么多，谢谢~

感谢郭神的文章~



8.4.4 Android动画合集之属性动画-又见

本节引言：

上节我们对Android的属性动画进行了初步的学习，相信大家对于属性动画已经不再是 一知半解的状态了，本节我们继续来探究Android属性动画的一些更高级的用法！ 依旧贴下郭神的三篇文章~

Android属性动画完全解析(上)，初识属性动画的基本用法

Android属性动画完全解析(中)，ValueAnimator和ObjectAnimator的高级用法

Android属性动画完全解析(下)，Interpolator和ViewPropertyAnimator的用法

内容依旧是参考的上述三篇文章，好的，开始本节内容~

1.Evaluator自定义

1）Evaluator介绍

上一节中的8.4.3 Android动画合集之属性动画-初见，使用动画的第一步都是：

调用ValueAnimator的**ofInt**()，**ofFloat**()或**ofObject**()静态方法创建ValueAnimator实例！

在例子中，ofInt和ofFloat我们都用到了，分别用于对浮点型和整型的数据进行动画操作！

那么**ofObject**()？初始对象和结束对象？如何过渡法？或者说这玩意怎么用？

好的，带着疑问，我们先来了解一个东西：Evaluator，在属性动画概念叨叨逼处其实我们就说到了这个东西：

Evaluator(计算器)

告诉动画系统如何从初始值过度到结束值，提供了下述几种Evaluator：
- IntEvaluator：用于计算int类型属性值的计算器
- FloatEvaluator：用于计算float类型属性值的计算器
- ArgbEvaluator：用于计算十六进制形式表示的颜色值的计算器
- TypeEvaluator：计算器的接口，我们可以实现该接口来完成自定义计算器

用来告诉动画系统如何从初始值过渡到结束值！好的，我们的入手点没错！ 我们进去IntEvaluator的源码，看下里面写了些什么？

```
public class IntEvaluator implements TypeEvaluator<Integer> {

    /**
     * This function returns the result of linearly interpolating the start and end
     * <code>fraction</code> representing the proportion between the start and end v
     * calculation is a simple parametric calculation: <code>result = x0 + t * (v1 -
     * where <code>x0</code> is <code>startValue</code>, <code>x1</code> is <code>en
     * and <code>t</code> is <code>fraction</code>.
     *
     * @param fraction   The fraction from the starting to the ending values
     * @param startValue The start value; should be of type <code>int</code> or
     *                   <code>Integer</code>
     * @param endValue   The end value; should be of type <code>int</code> or <code>
     * @return A linear interpolation between the start and end values, given the
     *         <code>fraction</code> parameter.
     */
    public Integer evaluate(float fraction, Integer startValue, Integer endValue) {
        int startInt = startValue;
        return (int)(startInt + fraction * (endValue - startInt));
    }
}
```

嗯，实现了**TypeEvaluator**接口，然后重写了**evaluate()**方法，参数有三个，依次是：

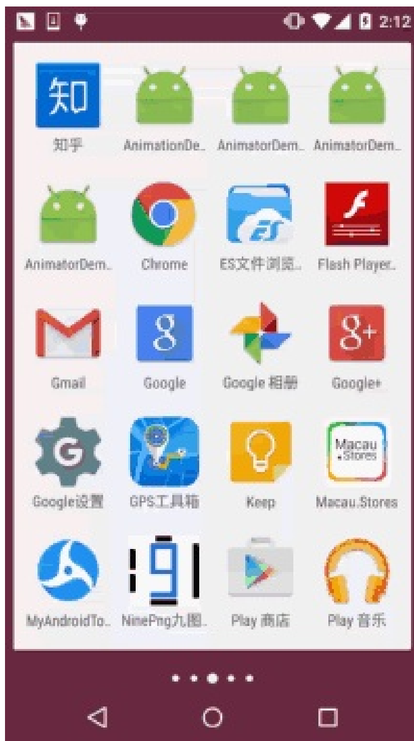
- **fraction**：动画的完成度，我们根据他来计算动画的值应该是多少
- **startValue**：动画的起始值
- **endValue**：动画的结束值

动画的值 **=** 初始值 **+** 完成度 ***** **(**结束值 **-** 初始值**)**

同样的还有FloatEvaluator，我们想告诉系统如何从初始对象过度到结束对象，那么我们就要 自己来实现**TypeEvaluator**接口，即自定义Evaluator了，说多无益，写个例子来看看：

2）使用示例

运行效果图：



代码实现：

定义一个对象**Point.java**，对象中只有x，y两个属性以及get，set方法~

```
/**
 * Created by Jay on 2015/11/18 0018.
 */
public class Point {

    private float x;
    private float y;

    public Point() {
    }

    public Point(float x, float y) {
        this.x = x;
        this.y = y;
    }

    public float getX() {
        return x;
    }

    public float getY() {
        return y;
    }

    public void setX(float x) {
        this.x = x;
    }

    public void setY(float y) {
        this.y = y;
    }
}
```

接着自定义Evaluator类：**PointEvaluator.java**，实现接口重写evaluate方法~

```
/**
 * Created by Jay on 2015/11/18 0018.
 */
public class PointEvaluator implements TypeEvaluator<Point>{
    @Override
    public Point evaluate(float fraction, Point startValue, Point e
        float x = startValue.getX() + fraction * (endValue.getX() -
        float y = startValue.getY() + fraction * (endValue.getY() -
        Point point = new Point(x, y);
        return point;
    }
}
```

然后自定义一个View类：**AnimView.java**，很简单~

```
/**
 * Created by Jay on 2015/11/18 0018.
 */
public class AnimView extends View {

    public static final float RADIUS = 80.0f;
    private Point currentPoint;
    private Paint mPaint;

    public AnimView(Context context) {
        this(context, null);
    }

    public AnimView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public AnimView(Context context, AttributeSet attrs, int defSty
        super(context, attrs, defStyleAttr);
    }

    private void init() {
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mPaint.setColor(Color.BLUE);
    }

    private void drawCircle(Canvas canvas){
        float x = currentPoint.getX();
        float y = currentPoint.getY();
        canvas.drawCircle(x, y, RADIUS, mPaint);
    }

    private void startAnimation() {
        Point startPoint = new Point(RADIUS, RADIUS);
        Point endPoint = new Point(getWidth() - RADIUS, getHeight()
        ValueAnimator anim = ValueAnimator.ofObject(new PointEvalu
        anim.addUpdateListener(new ValueAnimator.AnimatorUpdateList
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                currentPoint = (Point) animation.getAnimatedValue(
                invalidate();
            }
        });
        anim.setDuration(3000l);
        anim.start();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (currentPoint == null) {
            currentPoint = new Point(RADIUS, RADIUS);
```

```
                drawCircle(canvas);
                startAnimation();
            } else {
                drawCircle(canvas);
            }
        }
    }
```

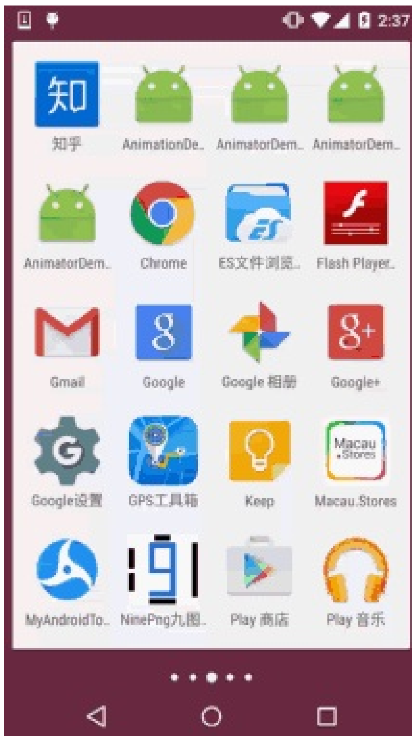
最后**MainActivity.java**处实例化这个View即可~

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new AnimView(this));
    }
}
```

3)示例增强版

我们上面示例的基础上加上圆移动时的颜色变化~ 这里我们另外用一个ObjectAnimator来加载颜色变化的动画，我们在View中加多个 int color来控制颜色，另外写上getColor()和setColor()的方法，我们先来自定义个Evaluator吧~

运行效果图：



实现代码：

ColorEvaluator.java：

```
/**
 * Created by Jay on 2015/11/18 0018.
 */
public class ColorEvaluator implements TypeEvaluator<Integer>{
    @Override
    public Integer evaluate(float fraction, Integer startValue, Int
        int alpha = (int) (Color.alpha(startValue) + fraction *
                (Color.alpha(endValue) - Color.alpha(startValue)))
        int red = (int) (Color.red(startValue) + fraction *
                (Color.red(endValue) - Color.red(startValue)));
        int green = (int) (Color.green(startValue) + fraction *
                (Color.green(endValue) - Color.green(startValue)))
        int blue = (int) (Color.blue(startValue) + fraction *
                (Color.blue(endValue) - Color.blue(startValue)));
        return Color.argb(alpha, red, green, blue);
    }
}
```

然后自定义View那里加个color，get和set方法；创建一个ObjectAnimator， 和AnimatorSet，接着把动画组合到一起就到，这里就加点东西而已，怕读者有问题， 直接另外建个View吧~

AnimView2.java：

```
/**
 * Created by Jay on 2015/11/18 0018.
 */
public class AnimView2 extends View {

    public static final float RADIUS = 80.0f;
    private Point currentPoint;
    private Paint mPaint;
    private int mColor;

    public AnimView2(Context context) {
        this(context, null);
    }

    public AnimView2(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public AnimView2(Context context, AttributeSet attrs, int defSt
        super(context, attrs, defStyleAttr);
    }

    private void init() {
```

```
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mPaint.setColor(Color.BLUE);
    }

    private void drawCircle(Canvas canvas){
        float x = currentPoint.getX();
        float y = currentPoint.getY();
        canvas.drawCircle(x, y, RADIUS, mPaint);
    }

    private void startAnimation() {
        Point startPoint = new Point(RADIUS, RADIUS);
        Point endPoint = new Point(getWidth() - RADIUS, getHeight()
        ValueAnimator anim = ValueAnimator.ofObject(new PointEvalua
        anim.addUpdateListener(new ValueAnimator.AnimatorUpdateList
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
                currentPoint = (Point) animation.getAnimatedValue()
                invalidate();
            }
        });

        ObjectAnimator objectAnimator = ObjectAnimator.ofObject(thi
                Color.BLUE, Color.RED);
        //动画集合将前面两个动画加到一起，with同时播放
        AnimatorSet animatorSet = new AnimatorSet();
        animatorSet.play(anim).with(objectAnimator);
        animatorSet.setStartDelay(1000l);
        animatorSet.setDuration(3000l);
        animatorSet.start();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (currentPoint == null) {
            currentPoint = new Point(RADIUS, RADIUS);
            drawCircle(canvas);
            startAnimation();
        } else {
            drawCircle(canvas);
        }
    }

    //color的get和set方法~
    public int getColor() {
        return mColor;
    }

    public void setColor(int color) {
        mColor = color;
        mPaint.setColor(color);
        invalidate();
    }
```

```
}
```

然后MainActivity，setContentView那里把AnimView改成AnimView2就好~

2.Interpolator(补间器)

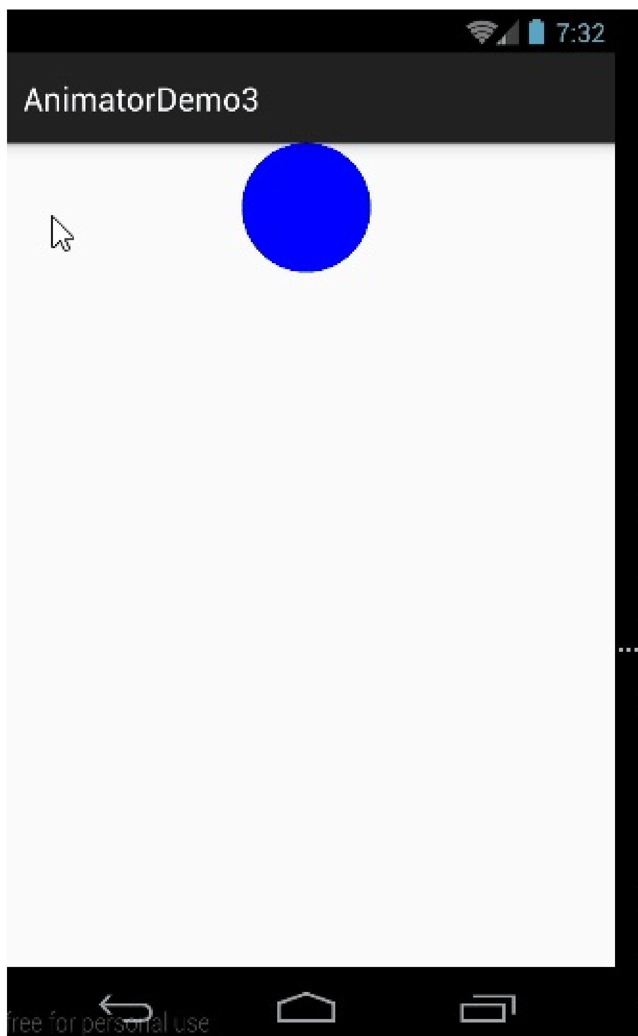
嗯，在讲补间动画的时候我们就讲过这个东东了~不知道你还有印象没？

在开始讲解各种动画的用法之前，我们先要来讲解一个东西：**Interpolator**

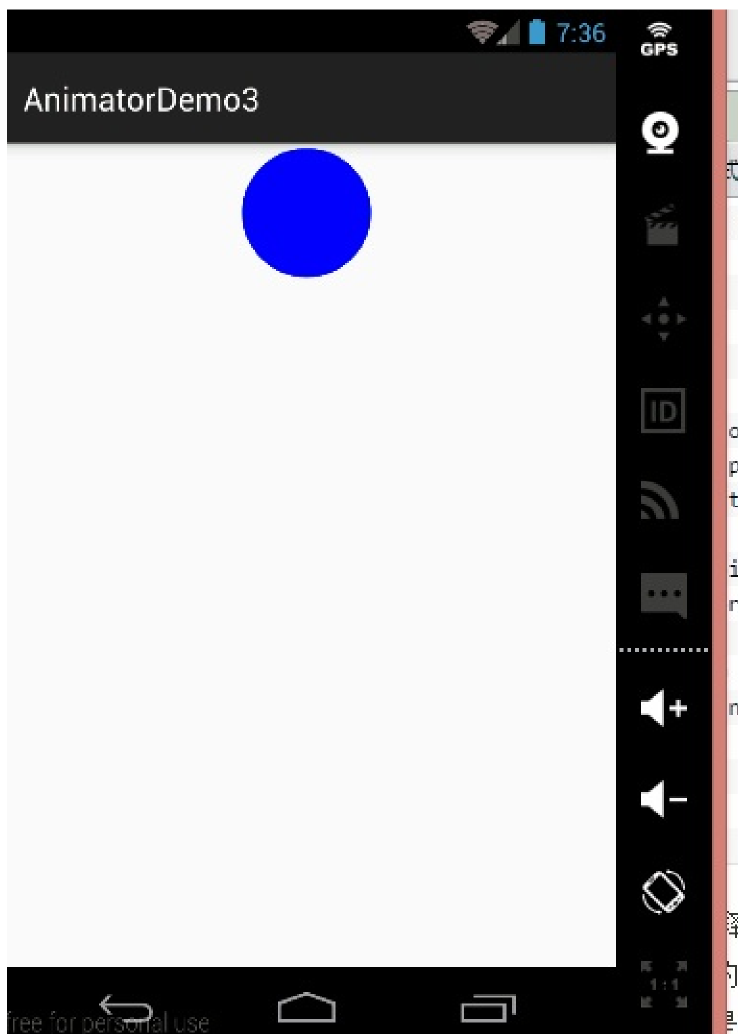
> 用来控制动画的变化速度，可以理解成动画渲染器，当然我们也可以自己实现Interpolator接口，自行来控制动画的变化速度，而Android中已经为我们提供了五个可供选择的实现类：
>
> - **LinearInterpolator**：动画以均匀的速度改变
> - **AccelerateInterpolator**：在动画开始的地方改变速度较慢，然后开始加速
> - **AccelerateDecelerateInterpolator**：在动画开始、结束的地方改变速度较慢，中间时加速
> - **CycleInterpolator**：动画循环播放特定次数，变化速度按正弦曲线改变：Math.sin(2 * mCycles * Math.PI * input)
> - **DecelerateInterpolator**：在动画开始的地方改变速度较快，然后开始减速
> - **AnticipateInterpolator**：反向，先向相反方向改变一段再加速播放
> - **AnticipateOvershootInterpolator**：开始的时候向后然后向前甩一定值后返回最后的值
> - **BounceInterpolator**： 跳跃，快到目的值时值会跳跃，如目的值100，后面的值可能依次为85，77，70，80，90，100
> - **OvershottInterpolator**：回弹，最后超出目的值然后缓慢改变到目的值
>
> 而这个东东，我们一般是在写动画xml文件时会用到，属性是：**android:interpolator**，而上面对应的值是：**@android:anim/linear_interpolator**，其实就是驼峰命名法变下划线而已 AccelerateDecelerateInterpolator对应：@android:anim/accelerate_decelerate_interpolator！

上面的补间器补间动画和属性动画都可用，而且补间动画还新增了一个**TimeInterpolator**接口 该接口是用于兼容之前的Interpolator的，这使得所有过去的Interpolator实现类都可以直接拿过来 放到属性动画当中使用！我们可以调用动画对象的setInterpolator()方法设置不同的Interpolator！ 我们先该点东西，让小球从屏幕正中央的顶部掉落到底部~ 然后我们会我们为我们的集合动画调用下述语句：**animatorSet.setInterpolator(new AccelerateInterpolator(2f));** 括号里的值用于控制加速度~

运行效果：



好像有点不和常理，正常应该是会弹起来的吧，我们换成**BounceInterpolator**试试~



嘿嘿，效果蛮赞的，当然还有N多个系统提供好的Interpolator，大家可以自己一一尝试，这里就 不慢慢跟大家纠结了~

下面我们来看看：

1）Interpolator的内部实现机制

我们先到TimeInterpolator接口的源码，发现这里只有一个**getInterpolation()**方法；

```
public interface TimeInterpolator {

    /**
     * Maps a value representing the elapsed fraction of an animation to a value that represents
     * the interpolated fraction. This interpolated value is then multiplied by the change in
     * value of an animation to derive the animated value at the current elapsed animation time.
     *
     * @param input A value between 0 and 1.0 indicating our current point
     *        in the animation where 0 represents the start and 1.0 represents
     *        the end
     * @return The interpolation value. This value can be more than 1.0 for
     *         interpolators which overshoot their targets, or less than 0 for
     *         interpolators that undershoot their targets.
     */
    float getInterpolation(float input);
}
```

简单的解释： getInterpolation()方法中接收一个input参数，这个参数的值会随着动画的运行而不断变化， 不过它的变化是非常有规律的，就是根据设定的动画时长匀速增加，变化范围是0到1。 也就是说当动画一开始的时候input的值是0，到动画结束的时候input的值是1，而中间的值则 是随着动画运行的时长在0到1之间变化的。

这里的**input**值决定了我们**TypeEvaluator**接口里的**fraction**的值。 input的值是由系统经过计算后传入到getInterpolation()方法中的，然后我们可以自己实现**getInterpolation**()方法中的算法，根据input的值来计算出一个返回值，而这个返回值就是fraction了。

我们可以看看**LinearInterpolator**里的代码：

```
/**
 * An interpolator where the rate of change is constant
 */
@HasNativeInterpolator
public class LinearInterpolator extends BaseInterpolator implements NativeInterpolatorFactory {

    public LinearInterpolator() {
    }

    public LinearInterpolator(Context context, AttributeSet attrs) {
    }

    public float getInterpolation(float input) { return input; }

    /** @hide */
    @Override
    public long createNativeInterpolator() {
        return NativeInterpolatorFactoryHelper.createLinearInterpolator();
    }
}
```

这里没有处理过直接返回input值，即fraction的值就是等于input的值，这就是匀速运动的 Interpolator的实现方式！其实无非就是算法不同，这就涉及到一些数学的东西了，又一次 体会到数学的重要性了，这里再贴个**BounceInterpolator**的源码吧：

```
/**
 * An interpolator where the change bounces at the end.
 */
@HasNativeInterpolator
public class BounceInterpolator extends BaseInterpolator implements NativeInterpolatorFactory {
    public BounceInterpolator() {
    }

    /UnusedDeclaration/
    public BounceInterpolator(Context context, AttributeSet attrs) {
    }

    private static float bounce(float t) { return t * t * 8.0f; }

    public float getInterpolation(float t) {
        // _b(t) = t * t * 8
        // bs(t) = _b(t) for t < 0.3535
        // bs(t) = _b(t - 0.54719) + 0.7 for t < 0.7408
        // bs(t) = _b(t - 0.8526) + 0.9 for t < 0.9644
        // bs(t) = _b(t - 1.0435) + 0.95 for t <= 1.0
        // b(t) = bs(t * 1.1226)
        t *= 1.1226f;
        if (t < 0.3535f) return bounce(t);
        else if (t < 0.7408f) return bounce(t - 0.54719f) + 0.7f;
        else if (t < 0.9644f) return bounce(t - 0.8526f) + 0.9f;
        else return bounce(t - 1.0435f) + 0.95f;
    }

    /** @hide */
    @Override
    public long createNativeInterpolator() {
        return NativeInterpolatorFactoryHelper.createBounceInterpolator();
    }
}
```

别问我这里的算法，我也不知道哈，我们再找个容易理解点的：**AccelerateDecelerateInterpolator**

```
import ...

/**
 * An interpolator where the rate of change starts and ends slowly but
 * accelerates through the middle.
 */
@HasNativeInterpolator
public class AccelerateDecelerateInterpolator extends BaseInterpolator
        implements NativeInterpolatorFactory {
    public AccelerateDecelerateInterpolator() {
    }

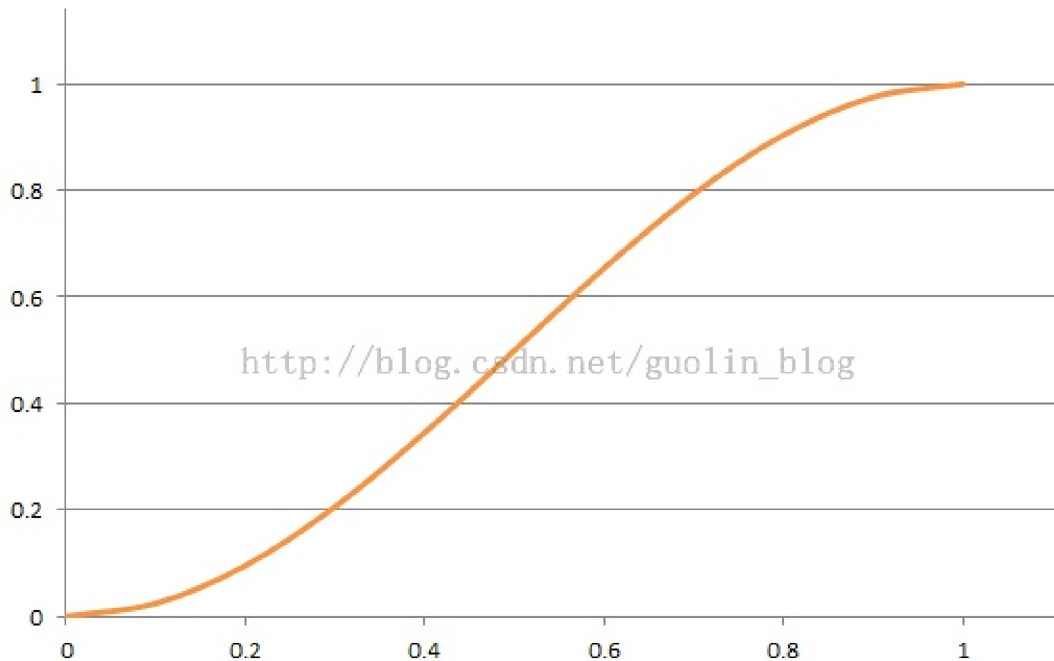
    /UnusedDeclaration/
    public AccelerateDecelerateInterpolator(Context context, AttributeSet attrs) {
    }

    public float getInterpolation(float input) {
        return (float)(Math.cos((input + 1) * Math.PI) / 2.0f) + 0.5f;
    }

    /** @hide */
    @Override
    public long createNativeInterpolator() {
        return NativeInterpolatorFactoryHelper.createAccelerateDecelerateInterpolator();
    }
}
```

这个Interpolator是先加速后减速效果的： **(float)(Math.cos((input + 1) * Math.PI) / 2.0f) + 0.5f** 的算法理解：

解：由input的取值范围为[0,1]，可以得出cos中的值的取值范围为[π,2π]，对应的值为-1和1；再用这个值来除以2加上0.5之后，getInterpolation()方法最终返回的结果值范围还是[0,1]，对应的曲线图如下：



所以是一个先加速后减速的过程！

嗯，学渣没法玩了...，上面全是郭大叔文章里搬过来的...我想静静...

2）自定义Interpolator

好吧，还是等会儿再忧伤吧，写个自定义的Interpolator示例先：非常简单，实现TimeInterpolator接口，重写getInterpolation方法

示例代码如下

```
private class DecelerateAccelerateInterpolator implements TimeInter
    @Override
    public float getInterpolation(float input) {
        if (input < 0.5) {
            return (float) (Math.sin(input * Math.PI) / 2);
        } else {
            return 1 - (float) (Math.sin(input * Math.PI) / 2);
        }
    }
}
```

调用setInterpolator(new DecelerateAccelerateInterpolator())设置下即可~ 限于篇幅就不贴图了~

3.ViewPropertyAnimator

3.1后系统当中附增的一个新的功能，为View的动画操作提供一种更加便捷的用法！ 假如是以前，让一个TextView从正常状态变成透明状态，会这样写：

```
ObjectAnimator animator = ObjectAnimator.ofFloat(textview, "alpha",
animator.start();
```

而使用ViewPropertyAnimator来实现同样的效果则显得更加易懂：

```
textview.animate().alpha(0f);
```

还支持连缀用法，组合多个动画，设定时长，设置Interpolator等~

```
textview.animate().x(500).y(500).setDuration(5000)
        .setInterpolator(new BounceInterpolator());
```

用法很简单，使用的时候查下文档就好~，另外下面有几个细节的地方要注意一下！

- 整个ViewPropertyAnimator的功能都是建立在View类新增的animate()方法之上的， 这个方法会创建并返回一个ViewPropertyAnimator的实例，之后的调用的所有方法， 设置的所有属性都是通过这个实例完成的。
- 使用ViewPropertyAnimator将动画定义完成之后，动画就会自动启动。 并且这个机制对于组合动画也同样有效，只要我们不断地连缀新的方法， 那么动画就不会立刻执行，等到所有在ViewPropertyAnimator上设置的方法都执行完毕后， 动画就会自动启动。当然如果不想使用这一默认机制的话，我们也可以显式地调用 **start()**方法来启动动画。

- ViewPropertyAnimator的所有接口都是使用连缀的语法来设计的，每个方法的返回值都是 它自身的实例，因此调用完一个方法之后可以直接连缀调用它的另一个方法，这样把所有的 功能都串接起来，我们甚至可以仅通过一行代码就完成任意复杂度的动画功能。

4.本节示例代码下载

AnimatorDemo3.zip

在Github上找到一个动画合集的项目，很多动画效果都有，下面贴下地址：

BaseAnimation 动画合集

想研究各种动画是如何实现的可自行查看源码~

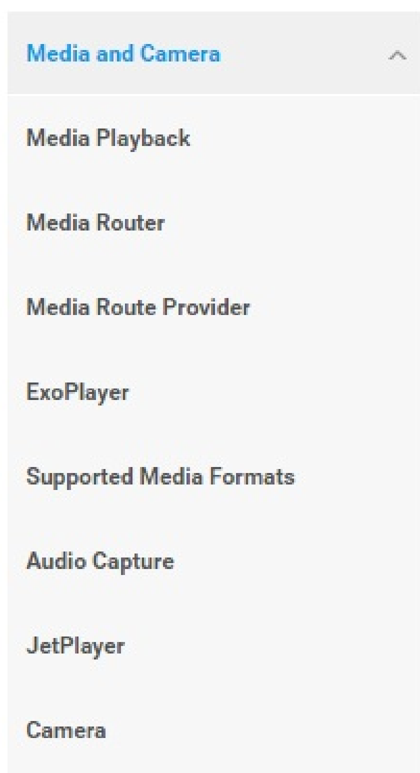
本节小结

> 嗯，本节我们讲了一些稍微高深一点的东西Evaluator啊，Interpolator啊，还有ViewPropertyAnimator，是不是又拓展了大家的见识~本节也是Android基础入门绘图 与的最后一小节了，如果大家把这一章节的内容都掌握了，再去学自定义控件， 或者看别人写的自定义控件，应该不会再那么地不知道从何入手，遇到一堆新面孔了吧！
>
> 嗯，还是谢谢郭神的文章，属性动画部分的内容很都是直接在郭神那里搬过来的  嘿嘿~本节就到这里，谢谢~
>
> PS:后面的示意图换模拟器是因为的N5秀逗了...

9.1 使用SoundPool播放音效(Duang~)

本节引言：

> 第九章给大家带来的是Android中的多媒体开发，与其说是多媒体开发还不如是多媒体相关API的 的使用，说下实际开发中我们做了一些和多媒体搭边的东西：拍照，录音，播放音乐，播放视频...
>
> 嗯，好吧，好像就这些了是吧，比如播放音乐，我们只是调用MediaPlayer，找到音乐文件， 然后调用下play方法播放而已...当然真正的多媒体开发又是另一个领域了，音视频的编码解码， 我等渣渣暂时只能仰望哈，我们知道怎么去调用这些API就好了！对了还是要科普下Android多媒体 框架的一些常识：
>
> 在Android上，预设的多媒体框架(multimedia framework)是**OpenCore**。OpenCore的优点是兼顾了 跨平台的移植性，而且已经过多方验证，所以相对来说较為稳定；但是其缺点是过於庞大复杂， 需要耗费相当多的时间去维护。而从Android 2.0开始，Google引进了架构稍微简洁一点的 **Stagefright**，当然没有完全抛弃OpenCore，主要是做了一个OMX层，仅仅是对OpenCore的omx-component部分做了引用。本来有逐渐取代OpenCORE的趋势，不过在今年八月份发现了 一个Stagefright漏洞，该漏洞允许远程代码执行，通过利用发送一个特制的MMS消息。
>
> 该漏洞对Android 2.2及更新版本均产生影响，对4.1及更新版本影响相对较弱。不明觉厉(都不知道在说什么JB)，嗯，好吧，科普完毕...这些东西知道下就好！
>
> 对了这个多媒体框架处于Android架构的第三层(**Libraries**)的**Media Framework**！ 另外如果你想知道Android这套多媒体框架支持什么类型的音视频数据可见官方文档：
>
> Supported Media Formats
>
> 你可以在这里直接点Media and Camera然后看下面的文档：



> 嗯，开头废话太多了，差点忘了今天的主角是**SoundPool**了，如题，SoundPool一般用来 播放密集，急促而又短暂的音效，比如特技音效：Duang~，游戏用得较多，你也可以为你的 APP添加上这个音效，比如酷狗音乐进去的时候播放"哈喽，酷狗"，其实这个创意还是不错的 间接的让用户知道了当前播放器的音量，不然用户一放歌，突然来了一发小苹果，引得附近 大妈起舞就不好了是吧；除了可以在音乐播放器加，你还可以在普通APP加上，比如收到推送 信息或者新的聊天信息，然后播放提示音，比如超级课程表新版本，加了这玩意，收到推送 信息会播放一段短促的"表表"的声音！SoundPool对象可以看作是一个可以从APK中导入资源 或者从文件系统中载入文件的样本集合。它利用MediaPlayer服务为音频解码为一个原始16位 PCM流。这个特性使得应用程序可以进行流压缩，而无须忍受在播放音频时解压所带来的**CPU** 负载和延时。SoundPool使用音效池的概念来管理多个播放流，如果超过流的最大数目， SoundPool会基于优先级自动停止先前播放的流，另外，SoundPool还支持自行设置声音的品质、 音量、 播放比率等参数。好了，话不多说，开始本节内容： 官方API文档：SoundPool

1.相关方法介绍：

1)构造方法：

> **SoundPool**(int maxStreams, int streamType, int srcQuality) 参数依次是：
>
> - ①指定支持多少个声音，SoundPool对象中允许同时存在的最大流的数量。
> - ②指定声音类型，流类型可以分为**STREAM_VOICE_CALL**, **STREAM_SYSTEM**, **STREAM_RING**,**STREAM_MUSIC** 和 **STREAM_ALARM**四种类型。在AudioManager中定义。
> - ③指定声音品质（采样率变换质量），一般直接设置为0！

在低版本中可以用上述构造方法，而API 21(Android 5.0)后这个构造方法就过时了！ 而用到一个SoundPool.Builder的东东，我们要实例化SoundPool只需调用：

```
SoundPool.Builder spb = new SoundPool.Builder();
spb.setMaxStreams(10);
spb.setAudioAttributes(null);    //转换音频格式
SoundPool sp = spb.build();      //创建SoundPool对象
```

要使用上述代码的话，TargetSDK版本要设置大于等于21哦！而且如果minSDK版本小于21 会出现下面的提醒：



2）常用方法介绍：

①加载声音资源：

> - **load**(Context context, int resId, int priority)
> - **load**(String path, int priority)
> - **load**(FileDescriptor fd, long offset, long length, int priority)
> - **load**(AssetFileDescriptor afd, int priority) 上述方法都会返回一个声音的ID，后面我们可以通过这个ID来播放指定的声音

参数介绍：

- context：上下文
- resId：资源id
- priority：没什么用的一个参数，建议设置为1，保持和未来的兼容性
- path：文件路径
- FileDescriptor：貌似是流吧，这个我也不知道
- AssetFileDescriptor：从asset目录读取某个资源文件，用法：**AssetFileDescriptor descriptor = assetManager.openFd("biaobiao.mp3")**；

②播放控制：

> **play**(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)

参数依次是：

- soundID：Load()返回的声音ID号
- leftVolume：左声道音量设置
- rightVolume：右声道音量设置
- priority：指定播放声音的优先级，数值越高，优先级越大。
- loop：指定是否循环：-1表示无限循环，0表示不循环，其他值表示要重复播放的次数
- rate：指定播放速率：1.0的播放率可以使声音按照其原始频率，而2.0的播放速率，可以使声音按照其 原始频率的两倍播放。如果为0.5的播放率，则播放速率是原始频率的一半。播放速率的取值范围是0.5至2.0。

③资源释放：

> 可以调用**release()**方法释放所有SoundPool对象占据的内存和资源，当然也可以根据声音 ID来释放！

3.使用代码示例：

运行效果图：



当点击按钮的时候会，"Duang"一下，这里演示了两种load的方法，分别是raw和assests！

关键代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_play1;
    private Button btn_play2;
    private Button btn_play3;
    private Button btn_play4;
    private Button btn_play5;
    private Button btn_release;
    private AssetManager aManager;
    private SoundPool mSoundPool = null;
    private HashMap<Integer, Integer> soundID = new HashMap<Integer

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        aManager = getAssets();
        try {
            initSP();
        } catch (Exception e) {
            e.printStackTrace();
        }
        bindViews();
    }

    private void bindViews() {
        btn_play1 = (Button) findViewById(R.id.btn_play1);
        btn_play2 = (Button) findViewById(R.id.btn_play2);
        btn_play3 = (Button) findViewById(R.id.btn_play3);
        btn_play4 = (Button) findViewById(R.id.btn_play4);
        btn_play5 = (Button) findViewById(R.id.btn_play5);
        btn_release = (Button) findViewById(R.id.btn_release);

        btn_play1.setOnClickListener(this);
        btn_play2.setOnClickListener(this);
        btn_play3.setOnClickListener(this);
        btn_play4.setOnClickListener(this);
        btn_play5.setOnClickListener(this);
        btn_release.setOnClickListener(this);

    }

    private void initSP() throws Exception{
        //设置最多可容纳5个音频流，音频的品质为5
        mSoundPool = new SoundPool(5, AudioManager.STREAM_SYSTEM, 5
```

```
        soundID.put(1, mSoundPool.load(this, R.raw.duang, 1));
        soundID.put(2 , mSoundPool.load(getAssets().openFd("biaobia
        soundID.put(3, mSoundPool.load(this, R.raw.duang, 1));
        soundID.put(4, mSoundPool.load(this, R.raw.duang, 1));
        soundID.put(5, mSoundPool.load(this, R.raw.duang, 1));
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_play1:
                mSoundPool.play(soundID.get(1), 1, 1, 0, 0, 1);
                break;
            case R.id.btn_play2:
                mSoundPool.play(soundID.get(2), 1, 1, 0, 0, 1);
                break;
            case R.id.btn_play3:
                mSoundPool.play(soundID.get(3), 1, 1, 0, 0, 1);
                break;
            case R.id.btn_play4:
                mSoundPool.play(soundID.get(4), 1, 1, 0, 0, 1);
                break;
            case R.id.btn_play5:
                mSoundPool.play(soundID.get(5), 1, 1, 0, 0, 1);
                break;
            case R.id.btn_release:
                mSoundPool.release();   //回收SoundPool资源
                break;
        }
    }
}
```

代码非常简单，另外如果你点击了最后一个按钮的话，SoundPool就会被释放，然后再其他按钮 就不会Duang了哦~

4.OnLoadCompleteListener监听声音文件是否加载完毕

> 嗯，这个是临时想起的，写完在写另一篇的时候突然想起，用法也很简单，我们可以 往上面的代码中添加OnLoadCompleteListener这个东东，然后重写onLoadComplete()方法， 最后为SoundPool对象设置这个东东即可！

```
mSoundPool.setOnLoadCompleteListener(new SoundPool.OnLoadCompleteLi
    @Override
    public void onLoadComplete(SoundPool soundPool, int sampleId, i
        Toast.makeText(MainActivity.this,"加特技准备完毕~",Toast.LENG
    }
});
```

5.示例代码下载：

SoundPoolDemo.zip

本节小结：

> 好的，本节给大家科普了一下Andorid多媒体的一些常识，以及教了大家如何为自己的APP添加音效， 只需通过简单的SoundPool即可实现，还等什么，往你的应用加上这个玩意，让你的应用Duang起来啊~
>
> 
>
> ，配合Demo食用更佳~

9.2 MediaPlayer播放音频与视频

本节引言：

> 本节带来的是Android多媒体中的——MediaPlayer，我们可以通过这个API来播放音频和视频 该类是Androd多媒体框架中的一个重要组件，通过该类，我们可以以最小的步骤来获取，解码 和播放音视频。它支持三种不同的媒体来源：
>
> - 本地资源
> - 内部的URI，比如你可以通过ContentResolver来获取
> - 外部URL(流) 对于Android所支持的的媒体格式列表
>
> 对于Android支持的媒体格式列表，可见：Supported Media Formats 文档
>
> 本节我们就来用MediaPlayer来写个简单的播放音视频的例子！
>
> 官方API文档：MediaPlayer

1.相关方法详解

1）获得MediaPlayer实例：

可以直接**new**或者调用**create**方法创建：

```
MediaPlayer mp = new MediaPlayer();
MediaPlayer mp = MediaPlayer.create(this, R.raw.test);  //无需再调用
```

另外create还有这样的形式： **create(Context context, Uri uri, SurfaceHolder holder)** 通过Uri和指定 SurfaceHolder 【抽象类】 创建一个多媒体播放器

2）设置播放文件：

```
//①raw下的资源：
MediaPlayer.create(this, R.raw.test);

//②本地文件路径：
mp.setDataSource("/sdcard/test.mp3");

//③网络URL文件：
mp.setDataSource("http://www.xxx.com/music/test.mp3");
```

> 另外setDataSource()方法有多个，里面有这样一个类型的参数：FileDescriptor，在使用这个 API的时候，需要把文件放到res文件夹平级的assets文件夹里，然后使用下述代码设置DataSource：

```
AssetFileDescriptor fileDescriptor = getAssets().openFd("rain.mp3"
m_mediaPlayer.setDataSource(fileDescriptor.getFileDescriptor(),file
```

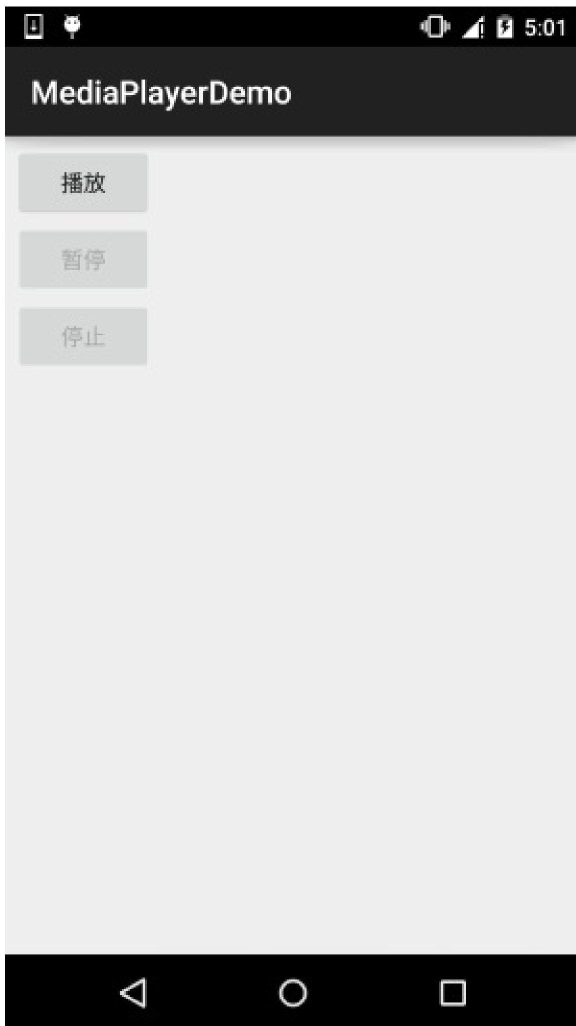
3）其他方法

> - **getCurrentPosition**( )：得到当前的播放位置
> - **getDuration**() ：得到文件的时间
> - **getVideoHeight**() ：得到视频高度
> - **getVideoWidth**() ：得到视频宽度
> - **isLooping**()：是否循环播放
> - **isPlaying**()：是否正在播放
> - **pause**()：暂停
> - **prepare**()：准备(同步)
> - **prepareAsync**()：准备(异步)
> - **release()**：释放MediaPlayer对象
> - **reset**()：重置MediaPlayer对象
> - **seekTo(int msec)**：指定播放的位置（以毫秒为单位的时间）
> - **setAudioStreamType(int streamtype)**：指定流媒体的类型
> - **setDisplay(SurfaceHolder sh)**：设置用SurfaceHolder来显示多媒体
> - **setLooping(boolean looping)**：设置是否循环播放
> - **setOnBufferingUpdateListener(MediaPlayer.OnBufferingUpdateListener listener)**： 网络流媒体的缓冲监听
> - **setOnCompletionListener(MediaPlayer.OnCompletionListener listener)**： 网络流媒体播放结束监听
> - **setOnErrorListener(MediaPlayer.OnErrorListener listener)**： 设置错误信息监听
> - **setOnVideoSizeChangedListener(MediaPlayer.OnVideoSizeChangedListener listener)**： 视频尺寸监听
> - **setScreenOnWhilePlaying(boolean screenOn)**：设置是否使用SurfaceHolder显示
> - **setVolume(float leftVolume, float rightVolume)**：设置音量
> - **start**()：开始播放
> - **stop**()：停止播放

2.使用代码示例

示例一：使用**MediaPlayer**播放音频：

运行效果图：



关键代码：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_play;
    private Button btn_pause;
    private Button btn_stop;
    private MediaPlayer mPlayer = null;
    private boolean isRelease = true;   //判断是否MediaPlayer是否释放

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_play = (Button) findViewById(R.id.btn_play);
        btn_pause = (Button) findViewById(R.id.btn_pause);
        btn_stop = (Button) findViewById(R.id.btn_stop);

        btn_play.setOnClickListener(this);
```

```
        btn_pause.setOnClickListener(this);
        btn_stop.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_play:
                if(isRelease){
                    mPlayer = MediaPlayer.create(this,R.raw.fly);
                    isRelease = false;
                }
                mPlayer.start();   //开始播放
                btn_play.setEnabled(false);
                btn_pause.setEnabled(true);
                btn_stop.setEnabled(true);
                break;
            case R.id.btn_pause:
                mPlayer.pause();     //停止播放
                btn_play.setEnabled(true);
                btn_pause.setEnabled(false);
                btn_stop.setEnabled(false);
                break;
            case R.id.btn_stop:
                mPlayer.reset();     //重置MediaPlayer
                mPlayer.release();   //释放MediaPlayer
                isRelease = true;
                btn_play.setEnabled(true);
                btn_pause.setEnabled(false);
                btn_stop.setEnabled(false);
                break;
        }
    }
}
```

注意事项：

> 播放的是res/raw目录下的音频文件，创建MediaPlayer调用的是create方法，第一次启动播放前 不需要再调用prepare()，如果是使用构造方法构造的话，则需要调用一次prepare()方法！ 另外贴下官方文档中，从其他两种途径播放音频的示例代码:

本地Uri：

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

外部**URL**：

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

> **Note**：假如你通过一个URL以流的形式播放在线音频文件，该文件必须可以进行 渐进式下载

示例二：使用**MediaPlayer**播放视频

MediaPlayer主要用于播放音频，没有提供图像输出界面，所以我们需要借助其他的 组件来显示MediaPlayer播放的图像输出，我们可以使用用**SurfaceView** 来显示，下面我们使用SurfaceView来写个视频播放的例子：

运行效果图：



实现代码：

布局文件：**activity_main.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <SurfaceView
        android:id="@+id/sfv_show"
        android:layout_width="match_parent"
        android:layout_height="300dp" />

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="开始" />

    <Button
        android:id="@+id/btn_pause"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="暂停 " />

    <Button
        android:id="@+id/btn_stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="终止" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Viev

    private MediaPlayer mPlayer = null;
    private SurfaceView sfv_show;
    private SurfaceHolder surfaceHolder;
    private Button btn_start;
    private Button btn_pause;
    private Button btn_stop;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }
```

```
    private void bindViews() {
        sfv_show = (SurfaceView) findViewById(R.id.sfv_show);
        btn_start = (Button) findViewById(R.id.btn_start);
        btn_pause = (Button) findViewById(R.id.btn_pause);
        btn_stop = (Button) findViewById(R.id.btn_stop);

        btn_start.setOnClickListener(this);
        btn_pause.setOnClickListener(this);
        btn_stop.setOnClickListener(this);

        //初始化SurfaceHolder类, SurfaceView的控制器
        surfaceHolder = sfv_show.getHolder();
        surfaceHolder.addCallback(this);
        surfaceHolder.setFixedSize(320, 220);   //显示的分辨率,不设置

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_start:
                mPlayer.start();
                break;
            case R.id.btn_pause:
                mPlayer.pause();
                break;
            case R.id.btn_stop:
                mPlayer.stop();
                break;
        }
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        mPlayer = MediaPlayer.create(MainActivity.this, R.raw.lesso
        mPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        mPlayer.setDisplay(surfaceHolder);    //设置显示视频显示在Surf
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, in

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {}

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPlayer.isPlaying()) {
            mPlayer.stop();
        }
        mPlayer.release();
```

```
        }
    }
```

代码很简单，布局有个SurfaceView，然后调用getHolder获得一个SurfaceHolder对象， 在这里完成SurfaceView相关的设置，设置了显示的分辨率以及一个Callback接口， 重写了SurfaceView创建时，发生变化时，以及销毁时的三个方法！然后按钮控制播放 以及暂停而已~

示例三：使用VideoView播放视频

除了使用MediaPlayer + SurfaceView播放视频的方式，我们还可以使用VideoView来直接 播放视频，我们稍微改点东西就可以实现视频播放！运行效果和上面的一致，就不贴了， 直接上代码！

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private VideoView videoView;
    private Button btn_start;
    private Button btn_pause;
    private Button btn_stop;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        videoView = (VideoView) findViewById(R.id.videoView);
        btn_start = (Button) findViewById(R.id.btn_start);
        btn_pause = (Button) findViewById(R.id.btn_pause);
        btn_stop = (Button) findViewById(R.id.btn_stop);

        btn_start.setOnClickListener(this);
        btn_pause.setOnClickListener(this);
        btn_stop.setOnClickListener(this);

        //根据文件路径播放
        if (Environment.getExternalStorageState().equals(Environmen
            videoView.setVideoPath(Environment.getExternalStorageDi
        }

        //读取放在raw目录下的文件
        //videoView.setVideoURI(Uri.parse("android.resource://com.j
        videoView.setMediaController(new MediaController(this));
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_start:
                videoView.start();
                break;
            case R.id.btn_pause:
                videoView.pause();
                break;
            case R.id.btn_stop:
                videoView.stopPlayback();
                break;
        }
    }
}
```

代码非常简单，就不解释了~有疑问的自己下个Demo运行下即可~

3.本节示例代码下载：

MediaPlayerDemo.zip

MediaPlayerDemo2.zip

VideoViewDemo.zip

本节小结：

> 好的，本节跟大家简单的介绍了下如何使用MediaPlayer播放音频以及结合SurfaceView 来播放视频，最后还写了一个用VideoView播放视频的例子，都是些非常简单的用法~ 相信大家学习起来非常简单~嗯，谢谢~

9.3 使用Camera拍照

本节引言

本节给大家带来的是Android中Camera的使用，简单点说就是拍照咯，无非两种：

1.调用系统自带相机拍照，然后获取拍照后的图片

2.要么自己写个拍照页面

本节我们来写两个简单的例子体验下上面的这两种情况~

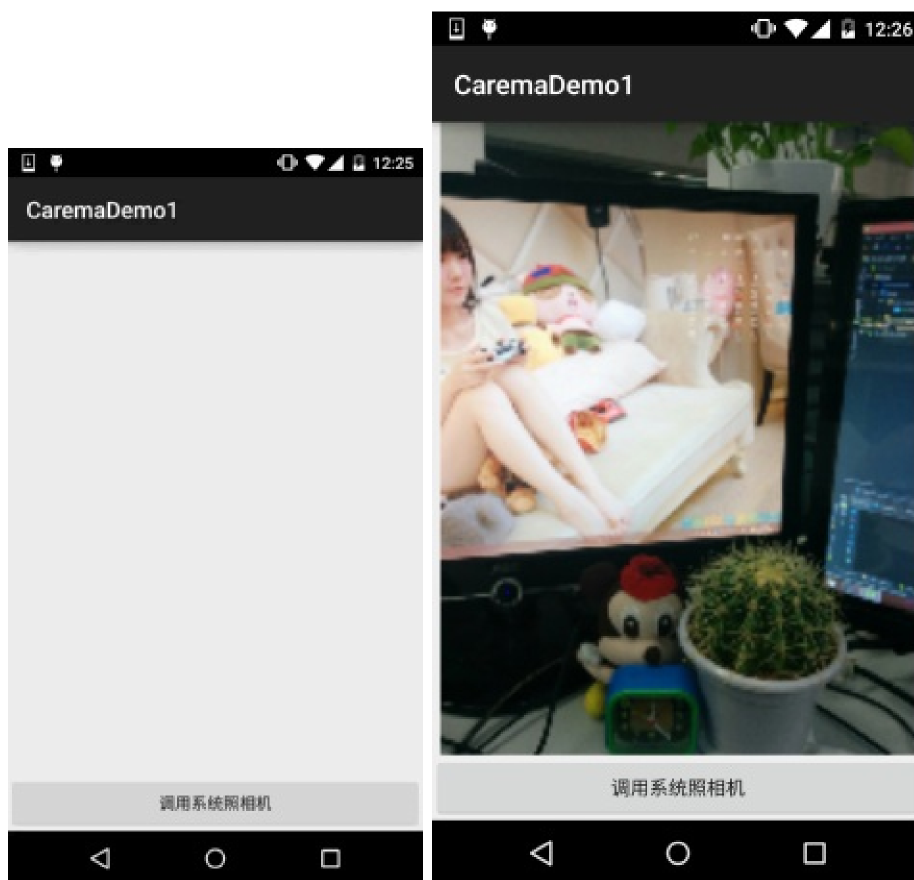
1.调用系统自带Carema

我们只需下面一席话语，即可调用系统相机，相机拍照后会返回一个intent给onActivityResult。 intent的extra部分包含一个编码过的Bitmap~

```
Intent it = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(it,Activity.DEFAULT_KEYS_DIALER);

//重写onActivityResult方法
    @Override
    protected void onActivityResult(int requestCode, int resultCode
        if(requestCode == Activity.RESULT_OK){
            Bundle bundle = data.getExtras();
            Bitmap bitmap = (Bitmap) bundle.get("data");
            img_show.setImageBitmap(bitmap);
        }
}
```

运行效果图：



这模糊的AV画质...毕竟是编码过后的Bitmap，对了，拍完的图片是不会保存到本地的， 我们可以自己写代码把图片保存到我们的SD卡里，然后再显示，这样的图片会清晰很多， 嗯，我们写代码来试下~

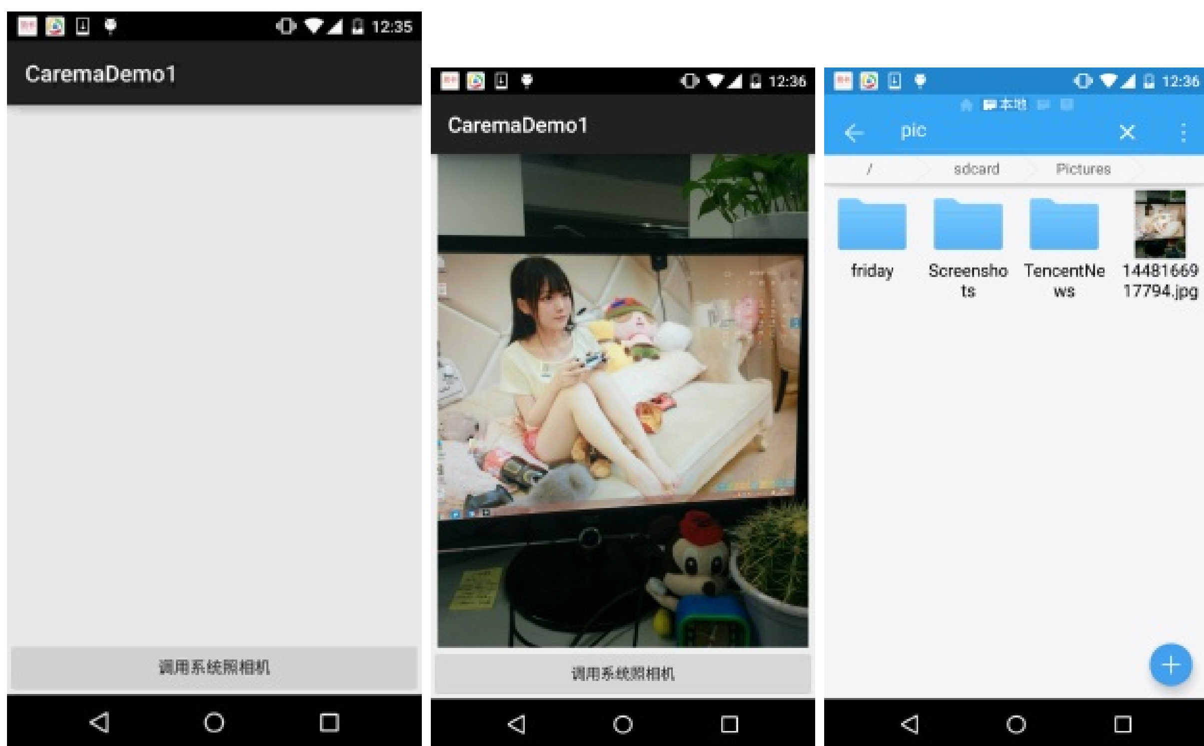
```
//定义一个保存图片的File变量
private File currentImageFile = null;

//在按钮点击事件处写上这些东西，这些是在SD卡创建图片文件的:
            @Override
            public void onClick(View v) {
                File dir = new File(Environment.getExternalStorageD
                if(dir.exists()){
                    dir.mkdirs();
                }
                currentImageFile = new File(dir,System.currentTimeM
                if(!currentImageFile.exists()){
                    try {
                        currentImageFile.createNewFile();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

                Intent it = new Intent(MediaStore.ACTION_IMAGE_CAPT
                it.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(c
                startActivityForResult(it, Activity.DEFAULT_KEYS_DI
            }

//onActivityResult：
 @Override
    protected void onActivityResult(int requestCode, int resultCode
        if (requestCode == Activity.DEFAULT_KEYS_DIALER) {
        img_show.setImageURI(Uri.fromFile(currentImageFile));
        }
}
```

好的，非常简单，我们来看下运行结果：

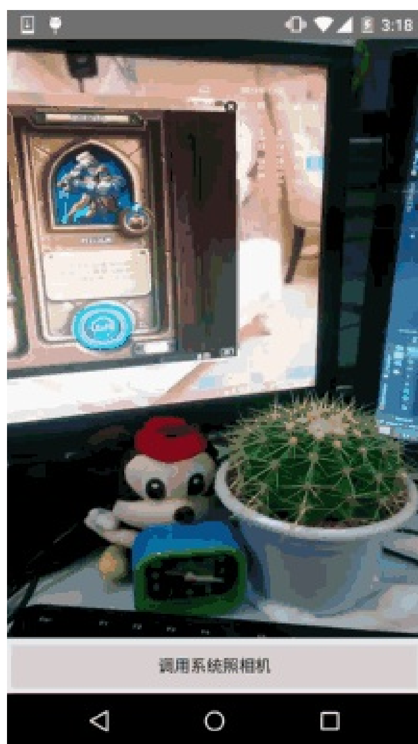


相比起上面那个清晰多了~调用系统自带Carema就是这么简单~

2.自己写一个拍照页面

这里我们需要用一个SurfaceView作为我们的预览界面，使用起来同一非常简单！

运行效果图：



代码实现：

布局代码：**activity_main.xml**：一个简单的surfaceView + Button

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <SurfaceView
        android:id="@+id/sfv_preview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1" />

    <Button
        android:id="@+id/btn_take"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="调用系统照相机" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private SurfaceView sfv_preview;
    private Button btn_take;
    private Camera camera = null;
    private SurfaceHolder.Callback cpHolderCallback = new SurfaceHo
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            startPreview();
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format

        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            stopPreview();
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
```

```
    }

    private void bindViews() {
        sfv_preview = (SurfaceView) findViewById(R.id.sfv_preview);
        btn_take = (Button) findViewById(R.id.btn_take);
        sfv_preview.getHolder().addCallback(cpHolderCallback);

        btn_take.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                camera.takePicture(null, null, new Camera.PictureCa
                    @Override
                    public void onPictureTaken(byte[] data, Camera
                        String path = "";
                        if ((path = saveFile(data)) != null) {
                            Intent it = new Intent(MainActivity.thi
                            it.putExtra("path", path);
                            startActivity(it);
                        } else {
                            Toast.makeText(MainActivity.this, "保存
                        }
                    }
                });
            }
        });
    }

    //保存临时文件的方法
    private String saveFile(byte[] bytes){
        try {
            File file = File.createTempFile("img","");
            FileOutputStream fos = new FileOutputStream(file);
            fos.write(bytes);
            fos.flush();
            fos.close();
            return file.getAbsolutePath();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return "";
    }

    //开始预览
    private void startPreview(){
        camera = Camera.open();
        try {
            camera.setPreviewDisplay(sfv_preview.getHolder());
            camera.setDisplayOrientation(90);   //让相机旋转90度
            camera.startPreview();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    //停止预览
    private void stopPreview() {
        camera.stopPreview();
        camera.release();
        camera = null;
    }

}
```

最后是另外一个PreviewActivity.java，这里将图片显示到界面上而已~

```
/**
 * Created by Jay on 2015/11/22 0022.
 */
public class PreviewActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ImageView img = new ImageView(this);
        String path = getIntent().getStringExtra("path");
        if(path != null){
            img.setImageURI(Uri.fromFile(new File(path)));
        }
        setContentView(img);
    }
}
```

嗯，都非常简单哈，别忘了加上权限：

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_S
```

另外，有一点要说的就是假如carema没有释放掉的话，那么下次调用carema就不会报错，报错内容是：java.lang.RuntimeException:fail to connect to camera service 所以，需要对Carema进行release();假如一直报上面的错误，请重启手机~

3.本节示例代码下载

CaremaDemo1.zip

CaremaDemo2.zip

本节小结

> 好的，本节给大家讲解了如何去调用系统自带相机获取拍照后的图片，以及自己写Carema来 完成自定义相机，嘿嘿，在某些场合下我们不需要拍照预览界面，我们直接把弄一个悬浮框， 然后点击悬浮框，就触发拍照事件，这不就可以实现什么不知鬼不觉的拍摄了么？(偷拍) 
>
> 嘿嘿，有点意思，要嗨自己动手写代码~

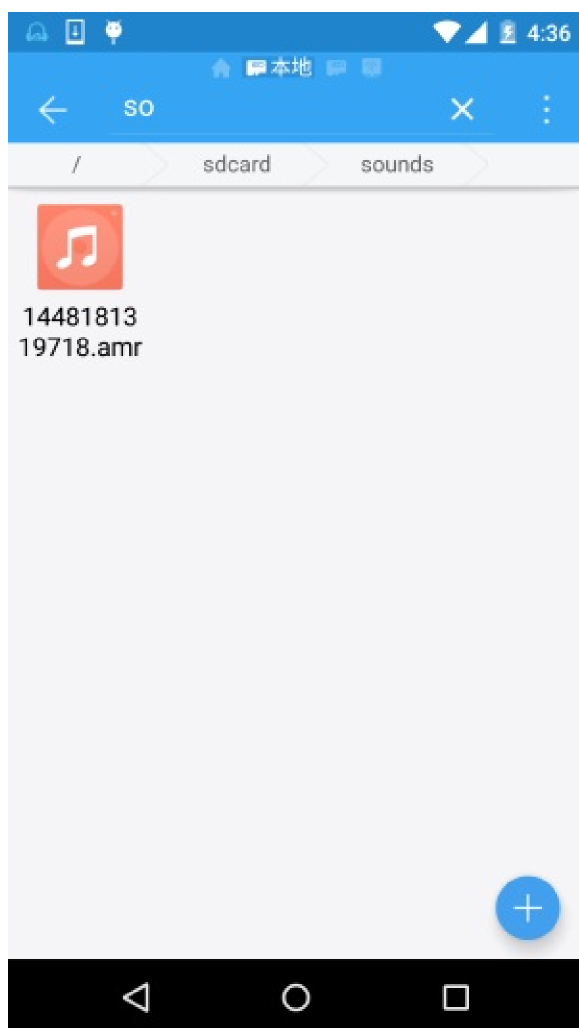
9.4 使用MediaRecord录音

本节引言

本节是Android多媒体基本API调用的最后一节，带来的是MediaRecord的简单使用，用法非常简单，我们写个例子来熟悉熟悉~

1.使用MediaRecord录制音频

运行结果：



实现代码：

布局代码：**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_control"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="开始录音" />

</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private Button btn_control;
    private boolean isStart = false;
    private MediaRecorder mr = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn_control = (Button) findViewById(R.id.btn_control);
        btn_control.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!isStart){
                    startRecord();
                    btn_control.setText("停止录制");
                    isStart = true;
                }else{
                    stopRecord();
                    btn_control.setText("开始录制");
                    isStart = false;
                }
            }
        });
    }

    //开始录制
    private void startRecord(){
        if(mr == null){
            File dir = new File(Environment.getExternalStorageDirec
            if(!dir.exists()){
                dir.mkdirs();
            }
```

```
                File soundFile = new File(dir,System.currentTimeMillis(
                if(!soundFile.exists()){
                    try {
                        soundFile.createNewFile();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                }
                mr = new MediaRecorder();
                mr.setAudioSource(MediaRecorder.AudioSource.MIC);  //音
                mr.setOutputFormat(MediaRecorder.OutputFormat.AMR_WB);
                mr.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_WB);
                mr.setOutputFile(soundFile.getAbsolutePath());
                try {
                    mr.prepare();
                    mr.start();  //开始录制
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }
        }

        //停止录制，资源释放
        private void stopRecord(){
            if(mr != null){
                mr.stop();
                mr.release();
                mr = null;
            }
        }
    }
```

最后别忘了在AndroidManifest.xml中添加下述权限：

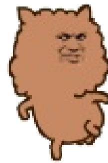
```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_ST
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

好的，就是这么简单~

2.本节示例代码下载

RecordDemo.zip

本节小结：

> 好的，本节内容非常简单，就是MediaRecorder的使用而已，大概是整套教程中最精简的一节 了吧~嘿嘿~ 

10.1 TelephonyManager(电话管理器)

本节引言：

> 本章节是Android基础入门教程的最后一章，主要讲解是一些零零散散的一些知识点，以及一些遗漏 知识点的补充，这些零散的知识点包括，各种系统服务的使用，比如本节的电话管理器，短信管理器， 振动器，闹钟，壁纸等等，还有传感器之类的东西！乱七八糟什么都有哈！好的，本节我们要学习的 是TelephonyManager，见名知义：用于管理手机通话状态，获取电话信息(设备信息、sim卡信息以及 网络信息)，侦听电话状态(呼叫状态服务状态、信号强度状态等)以及可以调用电话拨号器拨打电话！ 话不多开始本节内容~
>
> 官方API:TelephonyManager

1.获得TelephonyManager的服务对象

TelephonyManager tManager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);

2.用法示例

1)调用拨号器拨打电话号码

```
Uri uri=Uri.parse("tel:"+电话号码);
Intent intent=new Intent(Intent.ACTION_DIAL,uri);
startActivity(intent);
```

2)获取Sim卡信息与网络信息

运行效果图：



实现代码：

布局文件：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tv_phone1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />
```

```
    <TextView
        android:id="@+id/tv_phone3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/tv_phone8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/tv_phone9"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private TextView tv_phone1;
    private TextView tv_phone2;
    private TextView tv_phone3;
    private TextView tv_phone4;
    private TextView tv_phone5;
    private TextView tv_phone6;
    private TextView tv_phone7;
    private TextView tv_phone8;
    private TextView tv_phone9;
    private TelephonyManager tManager;
    private String[] phoneType = {"未知","2G","3G","4G"};
    private String[] simState = {"状态未知","无SIM卡","被PIN加锁","被P
            "被NetWork PIN加锁","已准备好"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //①获得系统提供的TelphonyManager对象的实例
        tManager = (TelephonyManager) getSystemService(Context.TELE
        bindViews();
    }

    private void bindViews() {
        tv_phone1 = (TextView) findViewById(R.id.tv_phone1);
        tv_phone2 = (TextView) findViewById(R.id.tv_phone2);
        tv_phone3 = (TextView) findViewById(R.id.tv_phone3);
        tv_phone4 = (TextView) findViewById(R.id.tv_phone4);
        tv_phone5 = (TextView) findViewById(R.id.tv_phone5);
        tv_phone6 = (TextView) findViewById(R.id.tv_phone6);
        tv_phone7 = (TextView) findViewById(R.id.tv_phone7);
        tv_phone8 = (TextView) findViewById(R.id.tv_phone8);
        tv_phone9 = (TextView) findViewById(R.id.tv_phone9);

        tv_phone1.setText("设备编号：" + tManager.getDeviceId());
        tv_phone2.setText("软件版本：" + (tManager.getDeviceSoftware
                tManager.getDeviceSoftwareVersion():"未知"));
        tv_phone3.setText("运营商代号：" + tManager.getNetworkOperato
        tv_phone4.setText("运营商名称：" + tManager.getNetworkOperato
        tv_phone5.setText("网络类型：" + phoneType[tManager.getPhone
        tv_phone6.setText("设备当前位置：" + (tManager.getCellLocatio
                .getCellLocation().toString() : "未知位置"));
        tv_phone7.setText("SIM卡的国别：" + tManager.getSimCountryIs
        tv_phone8.setText("SIM卡序列号：" + tManager.getSimSerialNum
        tv_phone9.setText("SIM卡状态：" + simState[tManager.getSimSt
    }
}
```

对了，别忘了在AndroidManifest.xml中加上权限哦！

```
<!-- 添加访问手机位置的权限 -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOC
<!-- 添加访问手机状态的权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE'
```

对了可能你想获取网络制式，而非普通的2G,3G,4G这样，其实我们可以到TelephonyManager类的源码里：

```
public static int getNetworkClass(int networkType) {
    switch (networkType) {
        case NETWORK_TYPE_GPRS:
        case NETWORK_TYPE_GSM:
        case NETWORK_TYPE_EDGE:
        case NETWORK_TYPE_CDMA:
        case NETWORK_TYPE_1xRTT:
        case NETWORK_TYPE_IDEN:
            return NETWORK_CLASS_2_G;
        case NETWORK_TYPE_UMTS:
        case NETWORK_TYPE_EVDO_0:
        case NETWORK_TYPE_EVDO_A:
        case NETWORK_TYPE_HSDPA:
        case NETWORK_TYPE_HSUPA:
        case NETWORK_TYPE_HSPA:
        case NETWORK_TYPE_EVDO_B:
        case NETWORK_TYPE_EHRPD:
        case NETWORK_TYPE_HSPAP:
        case NETWORK_TYPE_TD_SCDMA:
            return NETWORK_CLASS_3_G;
        case NETWORK_TYPE_LTE:
        case NETWORK_TYPE_IWLAN:
            return NETWORK_CLASS_4_G;
        default:
            return NETWORK_CLASS_UNKNOWN;
    }
}
```

我们可以根据这个networkType的值，判断不同的网络制式，比如，如果networkType == 1 那个是GPRS这种制式的~而这个networkType的值可以通过

```
public String getNetworkTypeName() { return getNetworkTypeName(getNetworkType()); }
```

即这个getNetworkType()方法获得！好了，就这么简单，可以像上面列好一个数组然后根据 不同的下标显示不同的值！ 对了，还有Sim卡状态的，字符串数组中的值，都可以到源码中看：

```
/**
 * Returns a constant indicating the state of the default SIM card.
 *
 * @see #SIM_STATE_UNKNOWN
 * @see #SIM_STATE_ABSENT
 * @see #SIM_STATE_PIN_REQUIRED
 * @see #SIM_STATE_PUK_REQUIRED
 * @see #SIM_STATE_NETWORK_LOCKED
 * @see #SIM_STATE_READY
 * @see #SIM_STATE_NOT_READY
 * @see #SIM_STATE_PERM_DISABLED
 * @see #SIM_STATE_CARD_IO_ERROR
 */
public int getSimState() {
    int slotIdx = getDefaultSim();
```

其他的可自行探索~

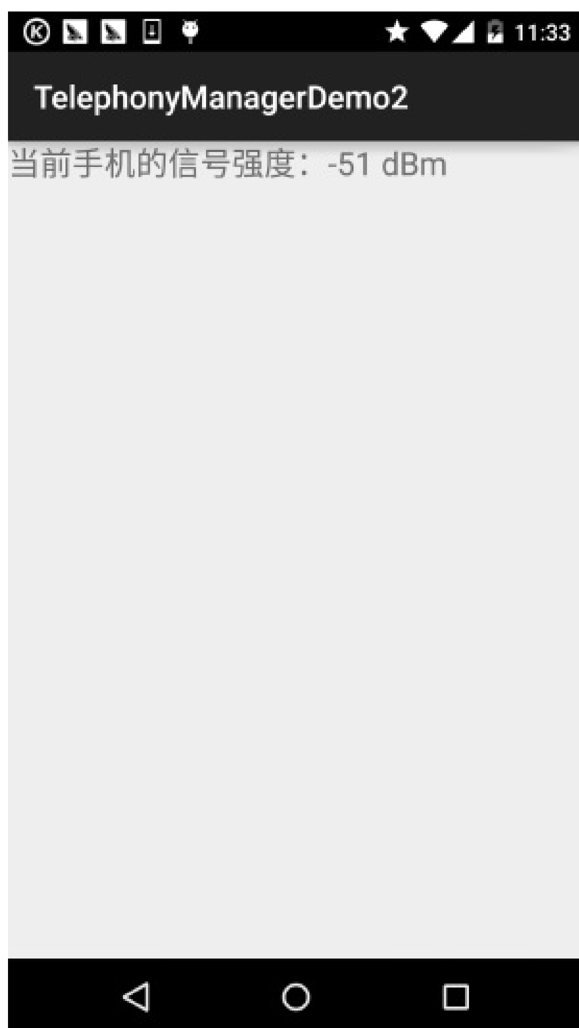
3)获取手机的信号强度

> 网络信号强度的单位是dBm(毫瓦分贝)，一般用负数表示，正常手机信号变化范围是从-110dBm (差)到-50dBm(好)之间，如果你比-50dBm还小的话，说明你就站在基站的附近，比如我的n5显示 的信号强度就是-51dBm，有时是-59dBm，因为隔壁就是南软大楼，上面就有基站...
>
> 另外2G，3G，4G获得信号强度的方式都是重写PhoneStateListener的onSignalStrengthsChanged() 方法，当信号强度发生改变的时候就会触发这个事件，我们可以在这个事件里获取信号强度！

手机获取信号**强**度代码示例：

dBm =-113+2*asu这是一个固定公式，asu(独立信号单元)

运行效果图：



实现代码：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private TextView tv_rssi;
    private MyPhoneStateListener mpsListener;
    private TelephonyManager tManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tManager = ((TelephonyManager)getSystemService(Context.TELE
        tv_rssi = (TextView) findViewById(R.id.tv_rssi);
        mpsListener  = new MyPhoneStateListener();
        tManager.listen(mpsListener,290);
    }

    private class MyPhoneStateListener extends PhoneStateListener {
        private int asu = 0,lastSignal = 0;
        @Override
        public void onSignalStrengthsChanged(SignalStrength signalS
            asu = signalStrength.getGsmSignalStrength();
            lastSignal = -113 + 2 * asu;
            tv_rssi.setText("当前手机的信号强度：" + lastSignal + " dB
            super.onSignalStrengthsChanged(signalStrength);
        }
    }
}
```

另外因为笔者的卡都是移动卡，联通和电信的不知道，但是从源码里看到这样几个API：

- **getEvdoDbm()**：电信3G
- **getCdmaDbm()**：联通3G
- **getLteDbm()**：4G

这些应该是可以直接获得dBm信号强度的，有条件的可以试试~

还有，别忘记加上权限了哦！

```
<!-- 添加访问手机状态的权限 -->
 <uses-permission android:name="android.permission.READ_PHONE_STATE
```

4)监听手机的所有来电

> 对于监听到的通话记录结果,你可以采取不同的方式获取到,这里用到的是把通话记录写入到文件中, 而你也可以以短信的形式发送给你,或者是上传到某个平台,当然如果通信记录不多的话还可以用短信 多了的话就很容易给人发现的了！另外,这里用的是Activity而非Service,就是说要打开这个Activity, 才可以进行监听,通常我们的需求都是要偷偷滴在后台跑的,因为时间关系就不写Service的了,如果需要 可自行修改,让Service随开机一起启动即可!

代码解析:

> 很简单,其实就是重写TelephonyManager的一个通话状态监听器**PhoneStateListener** 然后调用TelephonyManager.listen()的方法进行监听,当来电的时候, 程序就会将来电号码记录到文件中！

实现代码：

MainActivity.java：

```
public class MainActivity extends Activity
{
    TelephonyManager tManager;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // 取得TelephonyManager对象
        tManager = (TelephonyManager)
            getSystemService(Context.TELEPHONY_SERVICE);
        // 创建一个通话状态监听器
        PhoneStateListener listener = new PhoneStateListener()
        {
            @Override
            public void onCallStateChanged(int state, String number
            {
                switch (state)
                {
                // 无任何状态
                    case TelephonyManager.CALL_STATE_IDLE:
                        break;
                    case TelephonyManager.CALL_STATE_OFFHOOK:
                        break;
                    // 来电铃响时
                    case TelephonyManager.CALL_STATE_RINGING:
                        OutputStream os = null;
                        try
                        {
                            os = openFileOutput("phoneList", MODE_A
                        }
                        catch (FileNotFoundException e)
                        {
                            e.printStackTrace();
```

```
                            }
                            PrintStream ps = new PrintStream(os);
                            // 将来电号码记录到文件中
                            ps.println(new Date() + " 来电：" + number);
                            ps.close();
                            break;
                        default:
                            break;
                    }
                    super.onCallStateChanged(state, number);
                }
            };
            // 监听电话通话状态的改变
            tManager.listen(listener, PhoneStateListener.LISTEN_CALL_ST
    }
}
```

运行结果：

注意!要让这个程序位于前台哦!用另一个电话拨打该电话,接着就可以在DDMS的file Explorer的应用 对应包名的files目录下看到phoneList的文件了,我们可以将他导出到电脑中打开,文件的大概内容如下:

THR Oct 30 12:05:48 GMT 2014 来电: 137xxxxxxx

对了，别忘了权限！

```
<!-- 授予该应用读取通话状态的权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE'
```

5)黑名单来电自动挂断

> 所谓的黑名单就是将一些电话号码添加到一个集合中,当手机接收到这些电话的时候就直接挂断！ 但是Android并没有给我们提供挂断电话的API,于是乎我们需要通过AIDL来调用服务中的API来 实现挂断电话!
>
> 于是乎第一步要做的就是把android源码中的下面两个文件复制到src下的相应位置,他们分别是: com.android.internal.telephony包下的**ITelephony.aidl**;
>
> android.telephony包下的**NeighboringCellInfo.aidl**;
>
> 要创建对应的包哦!就是要把aidl文件放到上面的包下!!! 接着只需要调用ITelephony的endCall即可挂断电话!

这里给出的是简单的单个号码的拦截,输入号码,点击屏蔽按钮后,如果此时屏蔽的电话呼入的话; 直接会挂断,代码还是比较简单的,下面粘一下,因为用的模拟器是Genymotion,所以就不演示 程序运行后的截图了!

MainActivity.java：

```
public class MainActivity extends Activity {

    private TelephonyManager tManager;
    private PhoneStateListener pListener;
    private String number;
    private EditText locknum;
    private Button btnlock;

    public class PhonecallListener extends PhoneStateListener
    {
        @Override
        public void onCallStateChanged(int state, String incomingNu
            switch(state)
            {
            case TelephonyManager.CALL_STATE_IDLE:break;
            case TelephonyManager.CALL_STATE_OFFHOOK:break;
            //当有电话拨入时
            case TelephonyManager.CALL_STATE_RINGING:
                if(isBlock(incomingNumber))
                {
                    try
                    {
                        Method method = Class.forName("android.os.S
                                .getMethod("getService", String.cla
                        // 获取远程TELEPHONY_SERVICE的IBinder对象的代理
                        IBinder binder = (IBinder) method.invoke(nu
                            new Object[] { TELEPHONY_SERVICE });
                        // 将IBinder对象的代理转换为ITelephony对象
                        ITelephony telephony = ITelephony.Stub.asIn
                        // 挂断电话
                        telephony.endCall();
                    }catch(Exception e){e.printStackTrace();}
                }
                break;
            }
            super.onCallStateChanged(state, incomingNumber);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        locknum = (EditText) findViewById(R.id.locknum);
        btnlock = (Button) findViewById(R.id.btnlock);

        //获取系统的TelephonyManager管理器
        tManager = (TelephonyManager) getSystemService(TELEPHONY_SE
        pListener = new PhoneStateListener();
```

```
        tManager.listen(pListener, PhoneStateListener.LISTEN_CALL_S

        btnlock.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                number = locknum.getText().toString();
            }
        });

    }

    public boolean isBlock(String phone)
    {
        if(phone.equals(number))return true;
        return false;
    }
}
```

权限，权限，权限：

```
<!-- 授予该应用控制通话的权限 -->
<uses-permission android:name="android.permission.CALL_PHONE" />
<!-- 授予该应用读取通话状态的权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE'
```

另外，关于相关属性与方法中文版可见：Android电话信息相关API

3.本节示例代码下载

TelephonyManagerDemo.zip

TelephonyManagerDemo2.zip

黑名单拦截Demo.zip

本节小结：

> 好的，本节关于TelephonyManager(电话管理器)的学习就到这里，应该已经涵盖了 大部分的开发需求的了，如果有什么遗漏的，欢迎提出~
>
> 谢谢~

10.2 SmsManager(短信管理器)

本节引言：

> 本节带来的是Android中的SmsManager(短息管理器)，见名知意，就是用来管理手机短信的，而该类的应用场景并不多，一般是我们发短信的时候才会用到这个API，当然这种短信是 文字短信，对于彩信过于复杂，而且在QQ微信各种社交APP横行的年代，你会去发1块钱一条的 彩信吗？所以本节我们只讨论发送普通文字短信！ 官方文档：SmsManager

1.调用系统发送短信功能：

> 就是把写好的收信人和内容发送到系统的发送短信的界面，用户验证收件人内容是否真正确再点击发送! 说白了就是调用系统发短信的窗口,这样做有一定的好处:
>
> 这样发短信,app安装的时候就可以少写一条发短信的权限，那么诸如360这类安全软件在安装的时候 就不会提醒用户："这个APP有短信权限，可能会偷偷滴发短信喔"，而用户对于偷偷发短信的行为是十分 厌恶的，当然有些人不看直接安装，而有些人可能会觉得会偷偷发短信喔，好恶心的应用，我才不装咧， 又或者直接禁止我们的APP发送短信，那么当我们APP在发送短信的时候就可能会出现一些异常，或者 应用直接崩溃等！所以如果你的应用需要发送短信进行验证或者付费这些东西的话,建议使用这种方式!

核心代码：

```
public  void  SendSMSTo(String phoneNumber,String message){  //判断
```

2.调用系统提供的短信接口发送短信

> 这个就需要发短信的权限啦
>
> **uses-permission android:name="android.permission.SEND_SMS"/>**
>
> 我们直接调用SmsManager为我们提供的短信接口发送短信：
>
> **sendTextMessage**(destinationAddress, scAddress, text, sentIntent, deliverIntent);
>
> 参数依次是：
>
> - **destinationAddress**：收信人的电话号码
> - **scAddress**：短信中心的号码,null的话使用当前默认的短信服务中心
> - **text**：短信内容
> - **sentIntent**：短信发送状态的信息:(发送状态的Intent) 如果不为null，当消息成功发送或失败这个PendingIntent就广播。结果代码是Activity.RESULT_OK 表示成功，或RESULT_ERROR_GENERIC_FAILURE、RESULT_ERROR_RADIO_OFF、RESULT_ERROR_NULL_PDU 之一表示错误。对应RESULT_ERROR_GENERIC_FAILURE，sentIntent可能包括额外的"错误代码"包含一 个无线电广播技术特定的值，通常只在修复故障时有用。每一个基于SMS的应用程序控制检测sentIntent。 如果sentIntent是空，调用者将检测所有未知的应用程序，这将导致在检测的时候发送较小数量的SMS。
> - **deliverIntent**：短信是否被对方收到的状态信息:(接收状态的Intent) 如果不为null，当这个短信发送到接收者那里，这个PendtingIntent会被广播，状态报告生成的pdu（指对等层次之间传递的数据单位）会拓展到数据（"pdu"）

...那么复杂，pdu是什么卵？好吧，别纠结，简单知道这些参数是：

电话号码，信息中心，短信内容，是否发送成功的监听，以及收信人是否接受的监听就好了！

核心代码：

```
public  void sendSMS(String phoneNumber,String message){  //获取短信
    android.telephony.SmsManager smsManager = android.telephony.Sms
```

可能你还需要监听短信是否发送成功，或者收信人是否接收到信息，就把下面的加上吧：

1）处理返回发送状态的**sentIntent**

```
//处理返回的发送状态  String SENT_SMS_ACTION =  "SENT_SMS_ACTION";  Ir
```

2）处理返回接收状态的**deliverIntent**：

```
//处理返回的接收状态  String DELIVERED_SMS_ACTION =  "DELIVERED_SMS_A(
```

另外这里涉及到了广播的知识，如果你对广播不怎么了解的话，可以看下：

Android基础入门教程——BroadcastReceiver牛刀小试

Android基础入门教程——4.3.2 BroadcastReceiver庖丁解牛

本节小结：

>  好的，本节介绍了SmsManager发送文字短信的两种方式~非常简单~建议还是使用 第一种方案吧，起码用户体验好一点...

10.3 AudioManager(音频管理器)

本节引言：

> 在多媒体的第一节，我们用SoundPool写了个Duang的示例，小猪点击一个按钮后，突然发出"Duang"的 一声，而且当时的声音很大，吓死宝宝了，好在不是上班时间，上班时间偷偷写博客给经理知道 会作死的~嗯，好的，说到这个声音大小就得介绍下Android为我们提供的(音量大小控制)的API：
>
> **AudioManager**(音频管理器)了，该类位于Android.Media包下，提供了音量控制与铃声模式相关操作！ 本节我们就来学下这个东东的用法，你可以写一个Demo，一个简单的静音，每次看小电影之前，先 进Demo点下静音，然后
>
> ，说说而已哈~嗯，话不多说，开始本节内容！
>
> 官方API文档：AudioManager

1.获得AudioManager对象实例

> **AudioManager audiomanage = (AudioManager)context.getSystemService(Context.AUDIO_SERVICE);**

2.相关方法详解

常用方法：

> - **adjustVolume**(int direction, int flags)： 控制手机音量,调大或者调小一个单位,根据第一个参数进行判断 **AudioManager.ADJUST_LOWER**,可调小一个单位; **AudioManager.ADJUST_RAISE**,可调大一个单位
> - **adjustStreamVolume**(int streamType, int direction, int flags)： 同上,不过可以选择调节的声音类型 1）streamType参数,指定声音类型,有下述几种声音类型: **STREAM_ALARM**：手机闹铃 **STREAM_MUSIC**：手机音乐 **STREAM_RING**：电话铃声 **STREAM_SYSTEAM**：手机系统 **STREAM_DTMF**：音调 **STREAM_NOTIFICATION**：系统提示 **STREAM_VOICE_CALL**:语音电话 2）第二个参数和上面那个一样,调大或调小音量的 3）可选的标志位,比如AudioManager.**FLAG_SHOW_UI**,显示进度条,AudioManager.**PLAY_SOUND**:播放声音
> - **setStreamVolume**(int streamType, int index, intflags)：直接设置音量大小
> - **getMode**( )：返回当前的音频模式
> - **setMode**( )：设置声音模式 有下述几种模式: **MODE_NORMAL**(普通), **MODE_RINGTONE**(铃声), **MODE_IN_CALL**(打电话)，**MODE_IN_COMMUNICATION**(通话)
> - **getRingerMode**( )：返回当前的铃声模式
> - **setRingerMode**(int streamType):设置铃声模式 有下述几种模式: 如 **RINGER_MODE_NORMAL**（普通）、**RINGER_MODE_SILENT**（静音）、**RINGER_MODE_VIBRATE**（震动）
> - **getStreamVolume**(int streamType)： 获得手机的当前音量,最大值为7,最小值为0,当设置为0的时候,会自动调整为震动模式
> - **getStreamMaxVolume**(int streamType)：获得手机某个声音类型的最大音量值
> - **setStreamMute**(int streamType,boolean state)：将手机某个声音类型设置为静音
> - **setSpeakerphoneOn**(boolean on)：设置是否打开扩音器
> - **setMicrophoneMute**(boolean on)：设置是否让麦克风静音
> - **isMicrophoneMute**()：判断麦克风是否静音或是否打开
> - **isMusicActive**()：判断是否有音乐处于活跃状态
> - **isWiredHeadsetOn**()：判断是否插入了耳机

其他方法：

> - **abandonAudioFocus**(AudioManager.OnAudioFocusChangeListenerl)：放弃音频的焦点
> - **adjustSuggestedStreamVolume**(int,int suggestedStreamType intflags)： 调整最相关的流的音量，或者给定的回退流
> - **getParameters**(String keys)：给音频硬件设置一个varaible数量的参数值
> - **getVibrateSetting**(int vibrateType)：返回是否该用户的振动设置为振动类型
> - **isBluetoothA2dpOn**()：检查是否A2DP蓝牙耳机音频路由是打开或关闭
> - **isBluetoothScoAvailableOffCall**()：显示当前平台是否支持使用SCO的关闭调用用例
> - **isBluetoothScoOn**()：检查通信是否使用蓝牙SCO
> - **loadSoundEffects**()：加载声音效果
> - **playSoundEffect**((int effectType, float volume)：播放声音效果
> - **egisterMediaButtonEventReceiver**(ComponentName eventReceiver)：注册一个组件MEDIA_BUTTON意图的唯一接收机
> - **requestAudioFocus**(AudioManager.OnAudioFocusChangeListener l,int streamType,int durationHint) 请求音频的焦点
> - **setBluetoothScoOn**(boolean on)：要求使用蓝牙SCO耳机进行通讯
> - **startBluetoothSco/stopBluetoothSco()**()：启动/停止蓝牙SCO音频连接
> - **unloadSoundEffects**()：卸载音效

3.使用示例

> 嘿嘿，属性蛮多的，有些还涉及到蓝牙这些东东，这里我们只讲解最常见的一些方法！
>
> 遇到一些特殊的没见过的，我们再来查文档！
>
> 简单的示例：使用Mediaplayer播放音乐,通过AudioManager调节音量大小与静音！
>
> 对了，先在res下创建一个raw的文件夹，往里面丢一个MP3资源文件！

运行效果图：



代码实现：

布局代码**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="播放" />

    <Button
        android:id="@+id/btn_stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:enabled="false"
        android:text="停止" />

    <Button
        android:id="@+id/btn_higher"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="调高音量" />

    <Button
        android:id="@+id/btn_lower"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="调低音量" />

    <Button
        android:id="@+id/btn_quite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="静音" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_start;
    private Button btn_stop;
    private Button btn_higher;
    private Button btn_lower;
    private Button btn_quite;
    private MediaPlayer mePlayer;
    private AudioManager aManager;
```

```
    //定义一个标志用来标示是否点击了静音按钮
    private int flag = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获得系统的音频对象
        aManager = (AudioManager) getSystemService(Service.AUDIO_SE
        //初始化mediaplayer对象,这里播放的是raw文件中的mp3资源
        mePlayer = MediaPlayer.create(MainActivity.this, R.raw.coun
        //设置循环播放:
        mePlayer.setLooping(true);
        bindViews();
    }

    private void bindViews() {
        btn_start = (Button) findViewById(R.id.btn_start);
        btn_stop = (Button) findViewById(R.id.btn_stop);
        btn_higher = (Button) findViewById(R.id.btn_higher);
        btn_lower = (Button) findViewById(R.id.btn_lower);
        btn_quite = (Button) findViewById(R.id.btn_quite);

        btn_start.setOnClickListener(this);
        btn_stop.setOnClickListener(this);
        btn_higher.setOnClickListener(this);
        btn_lower.setOnClickListener(this);
        btn_quite.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_start:
                btn_stop.setEnabled(true);
                mePlayer.start();
                btn_start.setEnabled(false);
                break;
            case R.id.btn_stop:
                btn_start.setEnabled(true);
                mePlayer.pause();
                btn_stop.setEnabled(false);
                break;
            case R.id.btn_higher:
                // 指定调节音乐的音频，增大音量，而且显示音量图形示意
                aManager.adjustStreamVolume(AudioManager.STREAM_MUS
                        AudioManager.ADJUST_RAISE, AudioManager.FLA
                break;
            case R.id.btn_lower:
                // 指定调节音乐的音频，降低音量，只有声音,不显示图形条
                aManager.adjustStreamVolume(AudioManager.STREAM_MUS
                        AudioManager.ADJUST_LOWER, AudioManager.FLA
                break;
```

```
                case R.id.btn_quite:
                    // 指定调节音乐的音频，根据isChecked确定是否需要静音
                    flag *= -1;
                    if (flag == -1) {
                        aManager.setStreamMute(AudioManager.STREAM_MUSI
//                          aManager.adjustStreamVolume(AudioManager.STRE
//                                  AudioManager.FLAG_SHOW_UI);   //23以后
                        btn_quite.setText("取消静音");
                    } else {
                        aManager.setStreamMute(AudioManager.STREAM_MUSI
//                          aManager.adjustStreamVolume(AudioManager.STRE
//                                  AudioManager.FLAG_SHOW_UI);  //23以后
                        aManager.setMicrophoneMute(false);
                        btn_quite.setText("静音");
                    }
                    break;
            }
        }
    }
```

代码还是非常简单的，另外设置静音的方法**setStreamMute**()在API 23版本过期，可以使用另一个方法adjustStreamVolume(int, int, int)，然后第三个属性设置：

ADJUST_MUTE 或 ADJUST_UNMUTE！

对了，还有：

如果**adjustStreamVolume**()的第三个参数你设置了振动(Vibrator), 需要在AndroidManifest.xml中添加这个权限哦!

```
<**uses-permission android:name="android.permission.VIBRATE"**/>
```

4.代码示例下载

AudioManagerDemo.zip

本节小结：

> 好的，本节给大家演示了AudioManager用于调节音量的一个简单用法，这个类笔者也不常用 到，以后如果有什么新get的技能再加上吧~嘿嘿，静音Demo写好没？要结合实际需求哈~
>
> 另外，本周博客可能不会更新得太频繁，本周要把公司的WebSocket库替换掉，有得头痛了~ 好的，就说这么多，谢谢~

10.4 Vibrator(振动器)

本节引言：

> 本节我们介绍的是**Vibrator**(振动器)，是手机自带的振动器，别去百度直接搜针振动器，因为 你的搜索结果可能是如图所示的神秘的道具，或者其他神秘道具：
>
> 嗯，说回本节介绍的Vibrator，其实就是Android给我们提供的用于机身震动的一个服务！ 比如前面我们的Notification中可以设置震动，当收到推送消息的时候我们可以设置震动 提醒，游戏必备，比如"打飞机"的游戏，当你的飞机给人打爆的时候，会长震动！
>
> 下面我们就来写个简单的例子，来熟悉下这个Vibrator的用法！
>
> 官方API文档：Vibrator

1.获得Vibrator实例:

Vibrator vb = (Vibrator)getSystemService(Service.VIBRATOR_SERVICE);

2.可以使用的相关方法：

> - abstract void **cancel**()：关闭或者停止振动器
> - abstract boolean **hasVibrator**()：判断硬件是否有振动器
> - void **vibrate**(long milliseconds)：控制手机振动为milliseconds毫秒
> - void **vibrate**(long[] pattern,int repeat):指定手机以pattern指定的模式振动! 比如:pattern为new int[200,400,600,800],就是让他在200,400,600,800这个时间交替启动与关闭振动器! 而第二个则是重复次数,如果是-1的只振动一次,如果是0的话则一直振动 还有其他两个方法用得不多~ 对了，使用振动器还需要在AndroidManifest.xml中添加下述权限： **<uses-permission android:name="android.permission.VIBRATE"/>**

3.使用示例：设置频率不同的震动器：

> 对于Vibrator用的最广泛的莫过于所谓的手机按摩器类的app，在app市场一搜，一堆，笔者随便下了几个下来瞅瞅，都是大同小异的，这点小玩意竟然有8W多的下载量...好吧，好像也不算多，不过普遍功能都是切换振动频率来完成，而所谓的按摩效果，是否真的有效就不得而知了，那么接下来我们就来实现一个简单的按摩器吧！核心其实就是：vibrate()中的数组的参数,根据自己需求写一个数组就可以了！下述代码需要在真机上进行测试！

运行效果图：



实现代码：

简单的布局文件，五个按钮：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_hasVibrator"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="判断是否有振动器" />

    <Button
        android:id="@+id/btn_short"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="短振动" />

    <Button
        android:id="@+id/btn_long"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="长振动" />

    <Button
        android:id="@+id/btn_rhythm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="节奏振动" />

    <Button
        android:id="@+id/btn_cancle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="取消振动" />
</LinearLayout>
```

接着是**MainActivity.java**部分：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_hasVibrator;
    private Button btn_short;
    private Button btn_long;
    private Button btn_rhythm;
    private Button btn_cancle;
    private Vibrator myVibrator;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获得系统的Vibrator实例:
        myVibrator = (Vibrator) getSystemService(Service.VIBRATOR_S
        mContext = MainActivity.this;
        bindViews();
    }

    private void bindViews() {
        btn_hasVibrator = (Button) findViewById(R.id.btn_hasVibrato
        btn_short = (Button) findViewById(R.id.btn_short);
        btn_long = (Button) findViewById(R.id.btn_long);
        btn_rhythm = (Button) findViewById(R.id.btn_rhythm);
        btn_cancle = (Button) findViewById(R.id.btn_cancle);

        btn_hasVibrator.setOnClickListener(this);
        btn_short.setOnClickListener(this);
        btn_long.setOnClickListener(this);
        btn_rhythm.setOnClickListener(this);
        btn_cancle.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_hasVibrator:
                Toast.makeText(mContext, myVibrator.hasVibrator() ?
                        Toast.LENGTH_SHORT).show();
                break;
            case R.id.btn_short:
                myVibrator.cancel();
                myVibrator.vibrate(new long[]{100, 200, 100, 200},
                Toast.makeText(mContext, "短振动", Toast.LENGTH_SHOR
                break;
            case R.id.btn_long:
                myVibrator.cancel();
                myVibrator.vibrate(new long[]{100, 100, 100, 1000},
                Toast.makeText(mContext, "长振动", Toast.LENGTH_SHOR
                break;
            case R.id.btn_rhythm:
                myVibrator.cancel();
                myVibrator.vibrate(new long[]{500, 100, 500, 100, 5
                Toast.makeText(mContext, "节奏振动", Toast.LENGTH_SH
                break;
            case R.id.btn_cancle:
                myVibrator.cancel();
                Toast.makeText(mContext, "取消振动", Toast.LENGTH_SH
        }
    }
}
```

对了，别漏了振动器权限哦！

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

4.示例代码下载：

VibratorDemo.zip

本节小结：

> 好的，本节我们学习了Vibrator(振动器)的基本使用，代码非常简单，还不赶紧加入到 你的APP中，让你的应用HI起来~，嗯，就说这么多，谢谢，天色不早，小猪还是赶紧回家吧！ 毕竟我还是个黄花闺仔！万一湿身了就不好了~
>
> 

10.5 AlarmManager(闹钟服务)

本节引言：

> 本节带来的Android中的AlarmManager(闹钟服务)，听名字我们知道可以通过它开发手机闹钟类的APP， 而在文档中的解释是：在特定的时刻为我们广播一个指定的Intent， 简单说就是我们自己定一个时间， 然后当到时间时，AlarmManager会为我们广播一个我们设定好的Intent，比如时间到了，可以指向某个 Activity或者Service！另外官方文档中有一些要注意的地方：
>
> **Note: The Alarm Manager is intended for cases where you want to have your application code run at a specific time, even if your application is not currently running. For normal timing operations (ticks, timeouts, etc) it is easier and much more efficient to use `Handler`.**
>
> > **Note:** Beginning with API 19 (`KITKAT`) alarm delivery is inexact: the OS will shift alarms in order to minimize wakeups and battery use. There are new APIs to support applications which need strict delivery guarantees; see `setWindow(int, long, long, PendingIntent)` and `setExact(int, long, PendingIntent)` . Applications whose `targetSdkVersion` is earlier than API 19 will continue to see the previous behavior in which all alarms are delivered exactly when requested.
>
> You do not instantiate this class directly; instead, retrieve it through `Context.getSystemService(Context.ALARM_SERVICE)` .
>
> 另外要注意一点的是，AlarmManager主要是用来在某个时刻运行你的代码的，即时你的APP在那个特定 时间并没有运行！还有，从API 19开始，Alarm的机制都是非准确传递的，操作系统将会转换闹钟 ， 来最小化唤醒和电池的使用！某些新的API会支持严格准确的传递， 见 setWindow(int, long, long, PendingIntent)和setExact(int, long, PendingIntent)。 targetSdkVersion在API 19之前应用仍将继续使用以前的行为，所有的闹钟在要求准确传递的情况 下都会准确传递。更多详情可见官方API文档：AlarmManager

1.Timer类与AlarmManager类区别：

> 如果你学过J2SE的话，那么你对Timer肯定不会陌生，定时器嘛，一般写定时任务的时候 肯定离不开他，但是在Android里，他却有个短板，不太适合那些需要长时间在后台运行的 定时任务，因为Android设备有自己的休眠策略，当长时间的无操作，设备会自动让CPU进入 休眠状态，这样就可能导致Timer中的定时任务无法正常运行！而AlarmManager则不存在 这种情况，因为他具有唤醒CPU的功能，可以保证每次需要执行特定任务时CPU都能正常工作， 或者说当CPU处于休眠时注册的闹钟会被保留(可以唤醒CPU)，但如果设备被关闭，或者重新 启动的话，闹钟将被清除！(Android手机关机闹钟不响...)

2.获得AlarmManager实例对象：

```
AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_S
```

3.相关方法讲解：

> - **set**(int type,long startTime,PendingIntent pi)：一次性闹钟
> - **setRepeating**(int type，long startTime，long intervalTime, PendingIntent pi)： 重复性闹钟,和3有区别,3闹钟间隔时间不固定
> - **setInexactRepeating**（int type，long startTime，long intervalTime,PendingIntent pi）： 重复性闹钟，时间不固定
> - **cancel**(PendingIntent pi)：取消AlarmManager的定时服务
> - **getNextAlarmClock**()：得到下一个闹钟，返回值 AlarmManager.AlarmClockInfo
> - **setAndAllowWhileIdle**(int type, long triggerAtMillis, PendingIntent operation) 和set方法类似，这个闹钟运行在系统处于低电模式时有效
> - **setExact**(int type, long triggerAtMillis, PendingIntent operation)： 在规定的时间精确的执行闹钟，比set方法设置的精度更高
> - **setTime**(long millis)：设置系统墙上的时间
> - **setTimeZone**(String timeZone)：设置系统持续的默认时区
> - **setWindow**(int type, long windowStartMillis, long windowLengthMillis, PendingIntent operation)： 设置一个闹钟在给定的时间窗触发。类似于set，该方法允许应用程序精确地控制操作系统调 整闹钟触发时间的程度。

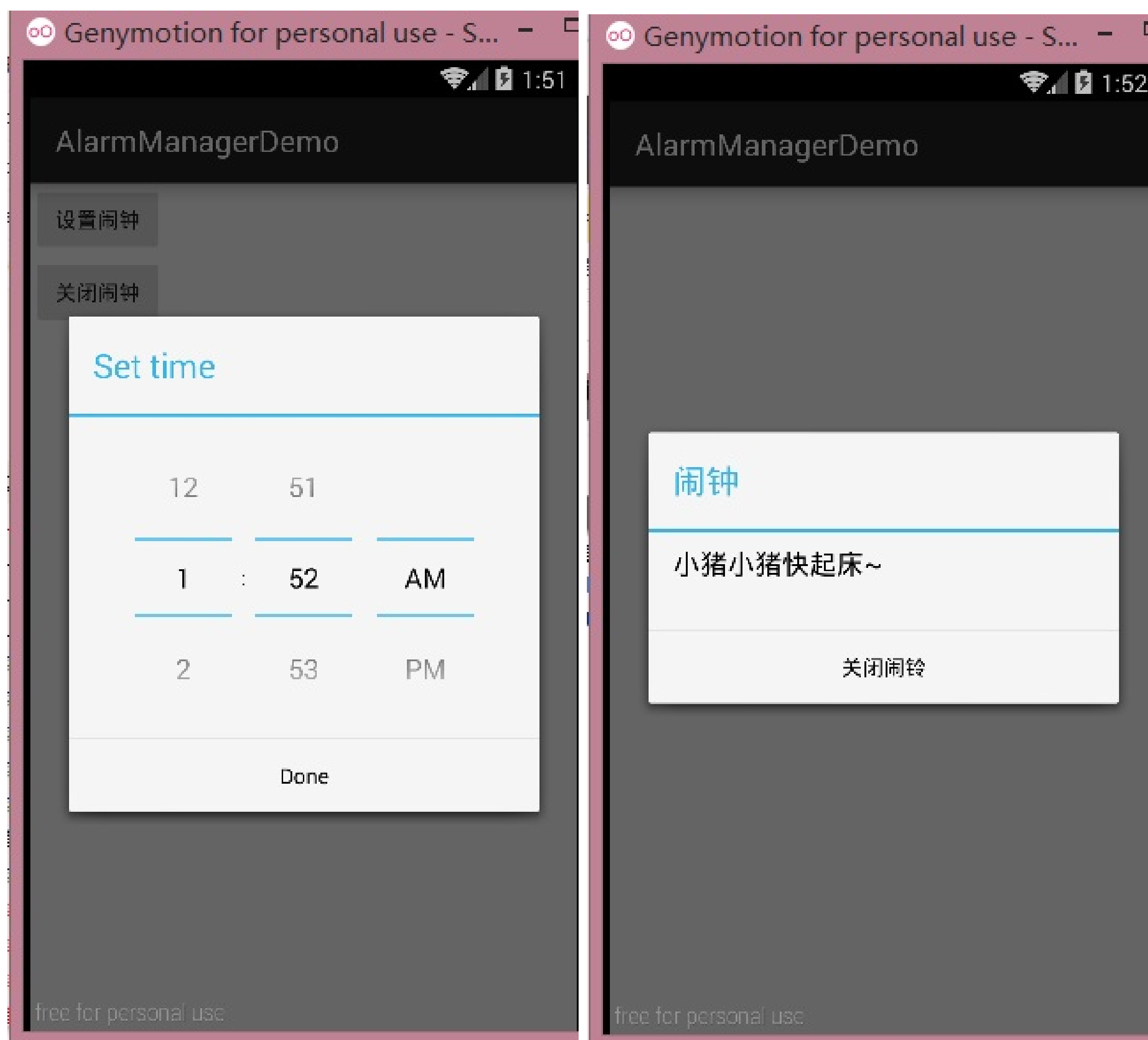
关键参数讲解：

> - **Type**(闹钟类型)： 有五个可选值: AlarmManager.**ELAPSED_REALTIME**: 闹钟在手机睡眠状态下不可用，该状态下闹钟使用相对时间（相对于系统启动开始），状态值为3; AlarmManager.**ELAPSED_REALTIME_WAKEUP** 闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟也使用相对时间，状态值为2； AlarmManager.**RTC** 闹钟在睡眠状态下不可用，该状态下闹钟使用绝对时间，即当前系统时间，状态值为1； AlarmManager.**RTC_WAKEUP** 表示闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟使用绝对时间，状态值为0; AlarmManager.**POWER_OFF_WAKEUP** 表示闹钟在手机关机状态下也能正常进行提示功能，所以是5个状态中用的最多的状态之一，该状态下闹钟也是用绝对时间，状态值为4；不过本状态好像受SDK版本影响，某些版本并不支持；
> - **startTime**：闹钟的第一次执行时间，以毫秒为单位，可以自定义时间，不过一般使用当前时间。 需要注意的是,本属性与第一个属性（type）密切相关,如果第一个参数对应的闹钟使用的是相对时间（ELAPSED_REALTIME和ELAPSED_REALTIME_WAKEUP），那么本属性就得使用相对时间 （相对于系统启动时间来说）,比如当前时间就表示为:SystemClock.elapsedRealtime()； 如果第一个参数对应的闹钟使用的是绝对时间(RTC、RTC_WAKEUP、POWER_OFF_WAKEUP）, 那么本属性就得使用绝对时间，比如当前时间就表示 为：System.currentTimeMillis()。
> - **intervalTime**：表示两次闹钟执行的间隔时间,也是以毫秒为单位.
> - **PendingIntent**：绑定了闹钟的执行动作，比如发送一个广播、给出提示等等。 PendingIntent是Intent的封装类。需要注意的是，如果是通过启动服务来实现闹钟提 示的话，PendingIntent对象的获取就应该采用Pending.getService (Context c,int i,Intent intent,int j)方法；如果是通过广播来实现闹钟 提示的话，PendingIntent对象的获取就应该采用PendingIntent.getBroadcast (Context c,int i,Intent intent,int j)方法；如果是采用Activity的方式来实 现闹钟提示的话，PendingIntent对象的获取就应该采用 PendingIntent.getActivity(Context c,int i,Intent intent,int j)方法。如果这三种方法错用了的话，虽然不会报错，但是看不到闹钟提示效果。

4.使用示例：一个简单的定时任务

> 要说的是，此例子只在Android 4.4以下的系统可行，5.0以上并不可行，后续如果有5.0 以上AlarmManager的解决方案，到时再补上！另外，这里用set方法可能有点不准，如果要 更精确的话可以使用setExtra()方法来设置AlarmManager!

运行效果图：



实现代码：

首先一个简单的布局文件:**activity_main.xml**，另外在res创建一个raw文件夹，把音频文件丢进去！另外创建一个只有外层布局的**activity_clock.xml**作为闹钟响时Activity的布局！没东西，就不贴了

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_set"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="设置闹钟" />

    <Button
        android:id="@+id/btn_cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="关闭闹钟"
        android:visibility="gone" />

</LinearLayout>
```

接着是**MainActivity.java**，也很简单：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_set;
    private Button btn_cancel;
    private AlarmManager alarmManager;
    private PendingIntent pi;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_set = (Button) findViewById(R.id.btn_set);
        btn_cancel = (Button) findViewById(R.id.btn_cancel);
        alarmManager = (AlarmManager) getSystemService(ALARM_SERVIC

        Intent intent = new Intent(MainActivity.this, ClockActivity
        pi = PendingIntent.getActivity(MainActivity.this, 0, intent

        btn_set.setOnClickListener(this);
        btn_cancel.setOnClickListener(this);

    }
```

```
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_set:
                Calendar currentTime = Calendar.getInstance();
                new TimePickerDialog(MainActivity.this, 0,
                        new TimePickerDialog.OnTimeSetListener() {
                            @Override
                            public void onTimeSet(TimePicker view,
                                                  int hourOfDay, ir
                                //设置当前时间
                                Calendar c = Calendar.getInstance(
                                c.setTimeInMillis(System.currentTir
                                // 根据用户选择的时间来设置Calendar对象
                                c.set(Calendar.HOUR, hourOfDay);
                                c.set(Calendar.MINUTE, minute);
                                // ②设置AlarmManager在Calendar对应的
                                alarmManager.set(AlarmManager.RTC_W
                                Log.e("HEHE",c.getTimeInMillis()+"'
                                // 提示闹钟设置完毕:
                                Toast.makeText(MainActivity.this, '
                                        Toast.LENGTH_SHORT).show();
                            }
                        }, currentTime.get(Calendar.HOUR_OF_DAY), c
                        .get(Calendar.MINUTE), false).show();
                btn_cancel.setVisibility(View.VISIBLE);
                break;
            case R.id.btn_cancel:
                alarmManager.cancel(pi);
                btn_cancel.setVisibility(View.GONE);
                Toast.makeText(MainActivity.this, "闹钟已取消", Toas
                        .show();
                break;
        }
    }
}
```

然后是闹铃页面的**ClockActivity.java**：

```
/**
 * Created by Jay on 2015/10/25 0025.
 */
public class ClockActivity extends AppCompatActivity {

    private MediaPlayer mediaPlayer;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_clock);
        mediaPlayer = mediaPlayer.create(this,R.raw.pig);
        mediaPlayer.start();
        //创建一个闹钟提醒的对话框,点击确定关闭铃声与页面
        new AlertDialog.Builder(ClockActivity.this).setTitle("闹钟"
                .setPositiveButton("关闭闹铃", new DialogInterface.C
                    @Override
                    public void onClick(DialogInterface dialog, int
                        mediaPlayer.stop();
                        ClockActivity.this.finish();
                    }
                }).show();
    }
}
```

代码非常简单，核心流程如下：

> - AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE); 获得系统提供的**AlarmManager**服务的对象
> - **Intent**设置要启动的组件: Intent intent = new Intent(MainActivity.this, ClockActivity.class);
> - **PendingIntent**对象设置动作,启动的是**Activity**还是**Service**,又或者是广播! PendingIntent pi = PendingIntent.getActivity(MainActivity.this, 0, intent, 0);
> - 调用**AlarmManager**的**set( )**方法设置单次闹钟的闹钟类型,启动时间以及**PendingIntent**对象! alarmManager.set(AlarmManager.RTC_WAKEUP,c.getTimeInMillis(), pi);

另外假如出现闹铃无效的话，你可以从这些方面入手：

> 1.系统版本或者手机，5.0以上基本没戏，小米，自行百度吧~ 2.ClockActivity有注册没？ 3.假如你用的是alarmManager发送广播，广播再激活Activity的话，则需要为Intent设置一个flag：i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK); 4.
>
> ```
> Intent intent = new Intent(MainActivity.this, ClockActivity.class);
> pi = PendingIntent.getActivity(MainActivity.this, 0, intent, 0);
> ```
>
> 这些地方没写错吧~别把getActivity写成了getService等哦~

另外，关于AlarmManager结合后来Service实现定时后台任务的例子，可见：4.2.2 Service进阶

5.代码示例下载：

AlarmManagerDemo.zip

本节小结：

> 好的，本节跟大家讲解了Android中的AlarmManager(闹钟服务)的使用，除了可以像例子那样定制 一个自己的闹钟，也可以结合Service，Thread来完成轮询等，用法多多，还需各位自行探究，嗯 本节就到这里，谢谢~ 

10.6 PowerManager(电源服务)

本节引言：

> 本节要讲解的是Android为我们提供的系统服务中的——**PowerManager**(电源服务)，用于 管理CPU运行，键盘或屏幕亮起来；不过，除非迫不得已，否则进来别去使用这个类，假如 你使用以后，一定要及时释放！本节并不会太深入滴去讲解这B，因为这涉及到底层的一些 东西，以后需要用到在深入研究~本节主要介绍的是一些基本的概念，PowerManager，wakelock 锁的机制等！
>
> 官方API文档：PowerManager

1.PowerManager是什么

> Android系统为我们提供的电源管理的一个API，其相关接口与设备电池的续航能力有很大的关联， 官方也说了，除非是迫不得已吧，不然的话，应该尽量避免使用这个类，并且使用完以后一定要及时释放!
>
> 所谓的电源管理包括:CPU运行，键盘或者屏幕亮起来!核心其实就是**wakelock**锁机制，只要我们拿着这个锁， 那么系统就无法进入休眠状态，可以给用户态程序或内核获取到！锁可以是:"有超时的"或者 "没有超时"，超时的锁到时间后会自动解锁，如果没有了锁或超时,内核会启动休眠机制来进入休眠!

2.wakelock锁介绍

> **PowerManager.WakeLock**有加锁与解锁两种状态，而加锁的形式有两种:
>
> ①永久锁住，这种锁除非显式的放开，否则是不会解锁的，所以用起来需要非常小心！
>
> ②超时锁，到时间后就会解锁，而创建WakeLock后，有两种加锁机制: ①不计数锁机制，②计数锁机制(默认) 可通过**setReferenceCounted**(boolean value)来指定,区别在于: 前者无论**acquire**( )多少次，一次**release**( )就可以解开锁。而后者则需要**(--count == 0)**的时候，同样当**(count == 0)**才会去申请锁 所以，**WakeLock**的计数机制并不是正真意义上对每次请求进行申请/释放一个锁; 只是对同一把锁被申请/释放的次数来进行统计，然后再去操作！

ps:关于更加深入的内容就涉及到底层的内容了，笔者水平有限，还没到那个level，这里就不深入研究了，就说一些基本的吧，以后有需要的话，再另开一篇吧~

3.PowerManager怎么用

```
PowerManager pm =  (PowerManager) getSystemService(Context.POWER_SE
```

上述**newWakeLock**( )的第一个**flag**标记，这些标记不同程度的影响系统电源.

这些标记都是独占的，并且每次只能指定其中一个。

PARTIAL_WAKE_LOCK:保持CPU 运转，屏幕和键盘灯有可能是关闭的。

SCREEN_DIM_WAKE_LOCK：保持CPU 运转，允许保持屏幕显示但有可能是灰的，允许关闭键盘灯

SCREEN_BRIGHT_WAKE_LOCK：保持CPU 运转，允许保持屏幕高亮显示，允许关闭键盘灯

FULL_WAKE_LOCK：保持CPU 运转，保持屏幕高亮显示，键盘灯也保持亮度

> ps:如果你使用的是局部唤醒锁的话（使用**PARTIAL_WAKE_LOCK**标志），CPU会继续运行， 将忽略任何的计时器，甚至按下电源按钮。其他的唤醒锁话，CPU也会继续运转，但是使用者仍 然可以按电源按钮让设备睡眠。另外，你可以使用两个以上的标记，但是他只影响屏幕的行为。 和**PARTIAL_WAKE_LOCK** 同时使用的话，没有任何影响。

屏幕解锁参数：

> **ACQUIRE_CAUSES_WAKEUP**：正常唤醒锁实际上并不打开照明。相反，一旦打开他们会一直仍然 保持(例如来世user的activity)。当获得wakelock，这个标志会使屏幕或/和键盘立即打开。
>
> 一个典型的使用就是可以立即看到那些对用户重要的通知。
>
> **ON_AFTER_RELEASE**：设置了这个标志，当wakelock释放时用户activity计时器会被重置，导致照明 持续一段时间。如果你在wacklock条件中循环，这个可以用来降低闪烁

4.需要的权限

要进行电源的操作需要在AndroidManifest.xml中声明该应用有设置电源管理的权限：

```
<uses-permission  android:name="android.permission.WAKE_LOCK"/>
```

你可能还需要：

```
<uses-permission  android:name="android.permission.DEVICE_POWER"/>
```

另外WakeLock的设置是**Activity**级别的，而不是针对整个Application应用的！

本节小结：

> 好的，本节介绍了PowerManager(电源服务)，不过仅仅是科普一下而已，内容也说了 不到迫不得已尽量别使用这个类~看懂了，或者没看懂都没关系，知道下即可！

10.7 WindowManager(窗口管理服务)

本节引言：

> 本节给大家带来的Android给我们提供的系统服务中的——WindowManager(窗口管理服务)， 它是显示View的最底层，Toast，Activity，Dialog的底层都用到了这个WindowManager， 他是全局的！该类的核心无非：调用addView，removeView，updateViewLayout这几个方法 来显示View以及通过WindowManager.LayoutParams这个API来设置相关的属性！
>
> 本节我们就来探讨下这个WindowManager在实际开发中的一些应用实例吧~
>
> 官方API文档：WindowManager

1.WindowManager的一些概念：

1）WindowManager介绍

> Android为我们提供的用于与窗口管理器进行交互的一个API！我们都知道App的界面都是 由一个个的Acitivty组成，而Activity又由View组成，当我们想显示一个界面的时候， 第一时间想起的是:Activity，对吧？又或者是Dialog和Toast。
>
> 但是有些情况下，前面这三者可能满足不了我们的需求，比如我们仅仅是一个简单的显示 用Activity显得有点多余了，而Dialog又需要Context对象，Toast又不可以点击... 对于以上的情况我们可以利用WindowManager这个东东添加View到屏幕上， 或者从屏幕上移除View！他就是管理Android窗口机制的一个接口，显示View的最底层！

2）如何获得WindowManager实例

①获得**WindowManager**对象:

```
WindowManager wManager = getApplicationContext().getSystemService((
```

②获得**WindowManager.LayoutParams**对象，为后续操作做准备

```
WindowManager.LayoutParams wmParams=new WindowManager.LayoutParams(
```

2.WindowManager使用实例：

实例1：获取屏幕宽高

在Android 4.2前我们可以用下述方法获得屏幕宽高：

```
public static int[] getScreenHW(Context context) {
    WindowManager manager = (WindowManager)context
    .getSystemService(Context.WINDOW_SERVICE);
    Display display = manager.getDefaultDisplay();
    int width = display.getWidth();
    int height = display.getHeight();
    int[] HW = new int[] { width, height };
    return HW;
}
```

而上述的方法在Android 4.2以后就过时了，我们可以用另一种方法获得屏幕宽高：

```
public static int[] getScreenHW2(Context context) {
    WindowManager manager = (WindowManager) context.
    getSystemService(Context.WINDOW_SERVICE);
    DisplayMetrics dm = new DisplayMetrics();
    manager.getDefaultDisplay().getMetrics(dm);
    int width = dm.widthPixels;
    int height = dm.heightPixels;
    int[] HW = new int[] { width, height };
    return HW;
}
```

然后我们可以再另外写两个获取宽以及高的方法，这里以第二种获得屏幕宽高为例：

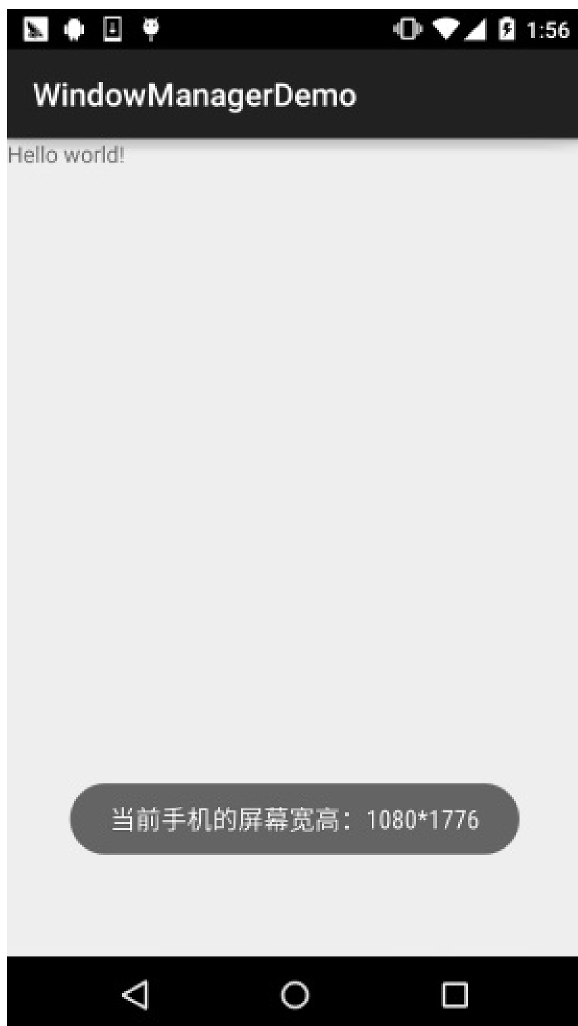
```
public static int getScreenW(Context context) {
    return getScreenHW2(context)[0];
}

public static int getScreenH(Context context) {
    return getScreenHW2(context)[1];
}
```

当然，假如你不另外写一个工具类的话，你可以直接直接获取，比如：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WindowManager wManager = (WindowManager) getSystemService(C
        DisplayMetrics dm = new DisplayMetrics();
        wManager.getDefaultDisplay().getMetrics(dm);
        Toast.makeText(MainActivity.this, "当前手机的屏幕宽高：" + dm.
                dm.heightPixels, Toast.LENGTH_SHORT).show();
    }
}
```

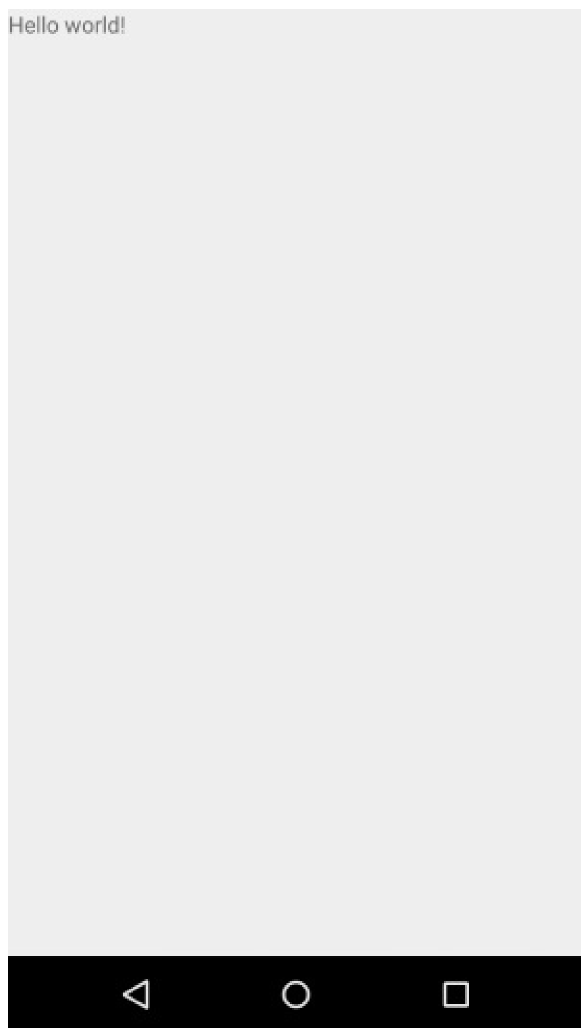
运行结果：



实例2：设置窗口全屏显示

```
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getSupportActionBar().hide();
```

运行结果：

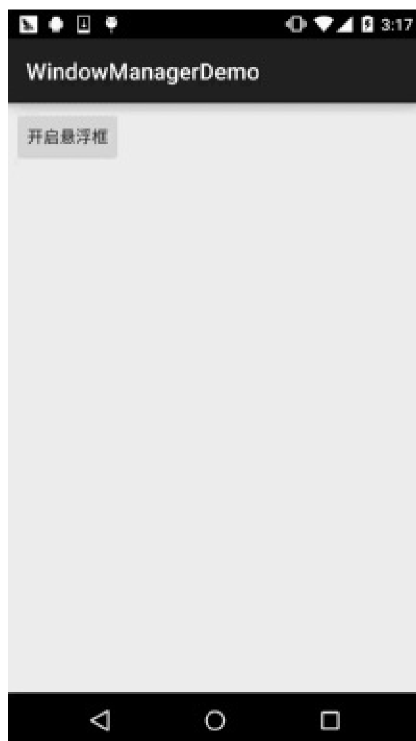


实例3：保持屏幕常亮

```
public void setKeepScreenOn(Activity activity,boolean keepScreenOn)
{
    if(keepScreenOn)
    {
        activity.getWindow().addFlags(WindowManager.LayoutParams.FL
    }else{
        activity.getWindow().clearFlags(WindowManager.LayoutParams.
    }
}
```

实例4：简单悬浮框的实现

运行效果图：



实现代码：

首先我们需要一个后台的Service在后台等待我们的操作，比如完成悬浮框的绘制移除等， 于是乎我们定义一个Service：**MyService.java**： 我们需要一个创建悬浮框View的一个方法：

```
private void createWindowView() {
    btnView = new Button(getApplicationContext());
    btnView.setBackgroundResource(R.mipmap.ic_launcher);
    windowManager = (WindowManager) getApplicationContext()
            .getSystemService(Context.WINDOW_SERVICE);
    params = new WindowManager.LayoutParams();

    // 设置Window Type
    params.type = WindowManager.LayoutParams.TYPE_SYSTEM_ALERT;
    // 设置悬浮框不可触摸
    params.flags = WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
            | WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE;
    // 悬浮窗不可触摸，不接受任何事件,同时不影响后面的事件响应
    params.format = PixelFormat.RGBA_8888;
    // 设置悬浮框的宽高
    params.width = 200;
    params.height = 200;
    params.gravity = Gravity.LEFT;
    params.x = 200;
    params.y = 000;
    // 设置悬浮框的Touch监听
    btnView.setOnTouchListener(new View.OnTouchListener() {
        //保存悬浮框最后位置的变量
        int lastX, lastY;
        int paramX, paramY;

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    lastX = (int) event.getRawX();
                    lastY = (int) event.getRawY();
                    paramX = params.x;
                    paramY = params.y;
                    break;
                case MotionEvent.ACTION_MOVE:
                    int dx = (int) event.getRawX() - lastX;
                    int dy = (int) event.getRawY() - lastY;
                    params.x = paramX + dx;
                    params.y = paramY + dy;
                    // 更新悬浮窗位置
                    windowManager.updateViewLayout(btnView, params)
                    break;
            }
            return true;
        }
    });
    windowManager.addView(btnView, params);
    isAdded = true;
}
```

然后我们只需在OnCreate( )方法中调用上述的createWindowView( )方法即可启动加载悬浮框， 但是我们发现了一点：这玩意貌似关不掉啊，卧槽，好吧，接下来我们就要分析下需求了!

当处于手机的普通界面，即桌面的时候，这玩意才显示，而当我们启动其他App时，这个悬浮框应该 消失不见,当我们推出app又回到桌面,这个悬浮框又要重新出现!

那么我们首先需要判断App是否位于桌面，于是乎我们再加上下述代码：

```
/**
 * 判断当前界面是否是桌面
 */
public boolean isHome(){
    if(mActivityManager == null) {
        mActivityManager = (ActivityManager)getSystemService(Contex
    }
    List<RunningTaskInfo> rti = mActivityManager.getRunningTasks(1)
    return homeList.contains(rti.get(0).topActivity.getPackageName(
}

/**
 * 获得属于桌面的应用的应用包名称
 * @return 返回包含所有包名的字符串列表
 */
private List<String> getHomes() {
    List<String> names = new ArrayList<String>();
    PackageManager packageManager = this.getPackageManager();
    // 属性
    Intent intent = new Intent(Intent.ACTION_MAIN);
    intent.addCategory(Intent.CATEGORY_HOME);
    List<ResolveInfo> resolveInfo = packageManager.queryIntentActiv
            PackageManager.MATCH_DEFAULT_ONLY);
    for(ResolveInfo ri : resolveInfo) {
        names.add(ri.activityInfo.packageName);
    }
    return names;
}
```

好了,接下来我们需要每隔一段时间来进行一系列的判断，比如：是否在桌面，是否已加载悬浮框， 否则加载；否则，如果加载了,就将这个悬浮框移除!这里我们使用handler~,因为不能在子线程直接 更新UI，所以，你懂的，所以我们自己写一个handler来完成上述的操作:

```
//定义一个更新界面的Handler
private Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch(msg.what) {
        case HANDLE_CHECK_ACTIVITY:
            if(isHome()) {
                if(!isAdded) {
                    windowManager.addView(btnView, params);
                    isAdded = true;
                new Thread(new Runnable() {
                    public void run() {
                        for(int i=0;i<10;i++){
                            try {
                                Thread.sleep(1000);
                            } catch (InterruptedException e) {e.pri
                            Message m = new Message();
                            m.what=2;
                            mHandler.sendMessage(m);
                        }
                    }
                }).start();}
            } else {
                if(isAdded) {
                    windowManager.removeView(btnView);
                    isAdded = false;
                }
            }
            mHandler.sendEmptyMessageDelayed(HANDLE_CHECK_ACTIVITY,
            break;
        }
    }
};
```

最后要做的一件事,就是重写Service的onStartCommand( )方法了,就是做判断,取出Intent中的 数据,判断是需要添加悬浮框,还是要移除悬浮框!

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    int operation = intent.getIntExtra(OPERATION, OPERATION_SHOW);
    switch(operation) {
    case OPERATION_SHOW:
        mHandler.removeMessages(HANDLE_CHECK_ACTIVITY);
        mHandler.sendEmptyMessage(HANDLE_CHECK_ACTIVITY);
        break;
    case OPERATION_HIDE:
        mHandler.removeMessages(HANDLE_CHECK_ACTIVITY);
        break;
    }
    return super.onStartCommand(intent, flags, startId);
}
```

好的，至此，主要的工作就完成了，接下来就是一些零碎的东西了，用一个Activity来启动这个Service：**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_on;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bindViews();
    }

    private void bindViews() {
        btn_on = (Button) findViewById(R.id.btn_on);
        btn_on.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_on:
                Intent mIntent = new Intent(MainActivity.this, Main
                mIntent.putExtra(MainService.OPERATION, MainService
                startService(mIntent);
                Toast.makeText(MainActivity.this, "悬浮框已开启~", To
                break;
        }
    }
}
```

接着AndroidManifest.xml加上权限，以及为MainService进行注册：

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WIND
<uses-permission android:name="android.permission.GET_TASKS" />

<service android:name=".MainService"/>
```

好了，逻辑还是比较容易理解的~大家自己再看看吧~

3.文献扩展：

> 从第四个实例中，你可能留意到了：WindowManager.LayoutParams这个东东，这是一个标记，比如全屏~时间关系就不一一列举出来了，可以到官网或者下述链接中查看：
>
> 官方文档：WindowManager.LayoutParams
>
> Android系统服务-WindowManager
>
> 另外，假如你对上述的悬浮框有兴趣，想更深入的研究，可见郭大叔(郭霖)的博客：
>
> Android桌面悬浮窗效果实现，仿360手机卫士悬浮窗效果
>
> Android桌面悬浮窗进阶，QQ手机管家小火箭效果实现

4.本节代码示例下载：

WindowManagerDemo2.zip

本节小结：

> 本节我们对Android系统服务中的WindowManager进行了学习，前面三个实例可能实际开发中会用得多一点，建议将第一个示例写成一个工具类，毕竟屏幕宽高用得蛮多的~至于悬浮框那个能看懂就看懂，没看懂耶没什么~实际开发很少叫你弄个悬浮框吧...嗯，好的，本节就到这里，谢谢~
>
> 

10.8 LayoutInflater(布局服务)

本节引言：

> 本节继续带来的是Android系统服务中的LayoutInflater(布局服务)，说到布局，大家第一时间 可能想起的是写完一个布局的xml，然后调用Activity的setContentView()加载布局，然后把他显示 到屏幕上是吧~其实这个底层走的还是这个LayoutInflater，用的Android内置的Pull解析器来解析 布局。一般在Android动态加载布局或者添加控件用得较多，本节我们就来学习下他在实际开发中 的一些用法~
>
> 官方**API**文档：LayoutInflater

1.LayoutInflater的相关介绍

1）Layout是什么鬼？

> 答：一个用于加载布局的系统服务，就是实例化与Layout XML文件对应的View对象，不能直接使用， 需要通过**getLayoutInflater**( )方法或**getSystemService**( )方法来获得与当前Context绑定的 LayoutInflater实例!

2）LayoutInflater的用法

①获取LayoutInflater实例的三种方法：

```
LayoutInflater inflater1 = LayoutInflater.from(this);
LayoutInflater inflater2 = getLayoutInflater();
LayoutInflater inflater3 = (LayoutInflater) getSystemService(LAYOUT
```

PS:后面两个其实底层走的都是第一种方法~

②加载布局的方法：

> public View **inflate** (int resource, ViewGroup root, boolean attachToRoot) 该方法的三个参数依次为:
>
> ①要加载的布局对应的资源id
>
> ②为该布局的外部再嵌套一层父布局，如果不需要的话，写null就可以了!
>
> ③是否为加载的布局文件的最外层套一层root布局，不设置该参数的话， 如果root不为null的话，则默认为true 如果root为null的话，attachToRoot就没有作用了! root不为null，attachToRoot为true的话，会在加载的布局文件最外层嵌套一层root布局; 为false的话，则root失去作用! 简单理解就是：是否为加载的布局添加一个**root**的外层容器**~!**

③通过LayoutInflater.LayoutParams来设置相关的属性:

> 比如RelativeLayout还可以通过addRule方法添加规则，就是设置位置：是参考父容器呢？ 还是参考子控件？又或者设置margin等等，这个由你决定~

2.纯Java代码加载布局

> 我们早已习惯了使用XML生成我们需要的布局，但是在一些特定的情况下，我们 需要使用Java代码往我们的布局中动态的添加组件或者布局！
>
> 但是不建议大家完全地使用Java代码来编写Android页面布局，首先一点就是代码会多， 一多久容易乱，而且不利于业务的分离，我们还是建议使用xml来完成布局，然后通过 Java代码对里面的组件进行修改，当然有些时候可能需要使用Java动态的来添加组件！

纯Java代码加载布局的流程：

——Step 1：

①创建容器:LinearLayout ly = new LinearLayout(this);

②创建组件:Button btnOne = new Button(this);

——Step 2:

可以为容器或者组件设置相关属性： 比如：**LinearLayout**，我们可以设置组件的排列方向：**ly.setOrientation(LinearLayout.VERTICAL);** 而组件也可以：比如Button：**btnOne.setText("按钮1");** 关于设置属性的方法可参见Android 的API,通常xml设置的属性只需在前面添加：set即可,比如 **setPadding**(左,上,右,下);

——Step 3：

将组件或容器添加到容器中，这个时候我们可能需要设置下组件的添加位置，或者设置他的大小： 我们需要用到一个类：LayoutParams，我们可以把它看成布局容器的一个信息包!封装位置与大小 等信息的一个类!先演示下设置大小的方法：(前面的LinearLayout可以根据不同容器进行更改)

```
LinearLayout.LayoutParams lp1 = new LinearLayout.LayoutParams(
        LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
```

很简单，接着就到这个设置位置了，设置位置的话，通常我们考虑的只是RelativeLayout! 这个时候用到LayoutParams的addRule( )方法!可以添加多个addRule( )哦！ 设置组件在父容器中的位置，

比如设置组件的对其方式:

```
RelativeLayout rly = new RelativeLayout(this);
RelativeLayout.LayoutParams lp2 = new RelativeLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTEN
lp2.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);
Button btnOne = new Button(this);
rly.addView(btnOne, lp2);
```

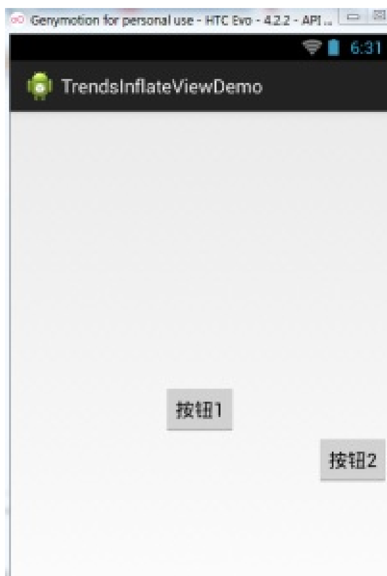
参照其他组件的对其方式：(有个缺点,就是要为参考组件手动设置一个id，是手动!!!!) 比如:设置btnOne居中后,让BtnTwo位于btnOne的下方以及父容器的右边!

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        RelativeLayout rly = new RelativeLayout(this);
        Button btnOne = new Button(this);
        btnOne.setText("按钮1");
        Button btnTwo = new Button(this);
        btnTwo.setText("按钮2");
        // 为按钮1设置一个id值
        btnOne.setId(123);
        // 设置按钮1的位置,在父容器中居中
        RelativeLayout.LayoutParams rlp1 = new RelativeLayout.Layou
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTEN
        rlp1.addRule(RelativeLayout.CENTER_IN_PARENT);
        // 设置按钮2的位置,在按钮1的下方,并且对齐父容器右面
        RelativeLayout.LayoutParams rlp2 = new RelativeLayout.Layou
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTEN
        rlp2.addRule(RelativeLayout.BELOW, 123);
        rlp2.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        // 将组件添加到外部容器中
        rly.addView(btnTwo, rlp2);
        rly.addView(btnOne, rlp1);
        // 设置当前视图加载的View即rly
        setContentView(rly);
    }
}
```

——step 4：

调用setContentView( )方法加载布局对象即可! 另外，如果你想移除某个容器中的View，可以调用容器.**removeView**(要移除的组件);

运行截图：



3.Java代码动态添加控件或xml布局

> 第二点我们讲解了使用纯Java代码来加载布局，实际当中用得并不多，更多的时候是动态 的添加View控件以及动态的加载XML布局！

1）Java代码动态增加View

动态添加组件的写法有两种，区别在于是否需要先**setContentView(R.layout.activity_main)**; 下面演示下两种不同写法添加一个Button的例子：

先写个布局文件先：**activity_main.xml**：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/txtTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="我是xml文件加载的布局"/>

</RelativeLayout>
```

第一种不需要**setContentView()**加载布局文件先：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Button btnOne = new Button(this);
        btnOne.setText("我是动态添加的按钮");
        RelativeLayout.LayoutParams lp2 = new RelativeLayout.Layout
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTEN
        lp2.addRule(RelativeLayout.CENTER_IN_PARENT);
        LayoutInflater inflater = LayoutInflater.from(this);
        RelativeLayout rly = (RelativeLayout) inflater.inflate(
                R.layout.activity_main, null)
                .findViewById(R.id.RelativeLayout1);
        rly.addView(btnOne,lp2);
        setContentView(rly);
    }
}
```

第二种不需要**setContentView()**加载布局文件先：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnOne = new Button(this);
        btnOne.setText("我是动态添加的按钮");
        RelativeLayout.LayoutParams lp2 = new RelativeLayout.Layout
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTEN
        lp2.addRule(RelativeLayout.CENTER_IN_PARENT);
        RelativeLayout rly = (RelativeLayout) findViewById(R.id.Rel
        rly.addView(btnOne,lp2);
    }
}
```

分析总结：

> 代码很简单,创建按钮后，我们又创建了一个LayoutParams对象，用来设置Button的大小， 又通过addRule()方法设置了Button的位置!
>
> 第一种方法：通过LayoutInflate的inflate()方法加载了activity_main布局，获得了外层容器， 接着addView添加按钮进容器,最后setContentView();
>
> 第二种方法：因为我们已经通过setContetView()方法加载了布局，此时我们就可以通过 findViewById找到这个外层容器,接着addView,最后setContentView()即可!
>
> 另外,关于这个setContentView( )他设置的视图节点是整个XML的根节点!

2）Java代码动态加载xml布局

接下来的话，我们换一个，这次加载的是xml文件!动态地添加**xml**文件! 先写下主布局文件和动态加载的布局文件：

activity_main.xml：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/btnLoad"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="动态加载布局"/>
</RelativeLayout>
```

inflate.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:id="@+id/ly_inflate" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="我是Java代码加载的布局" />

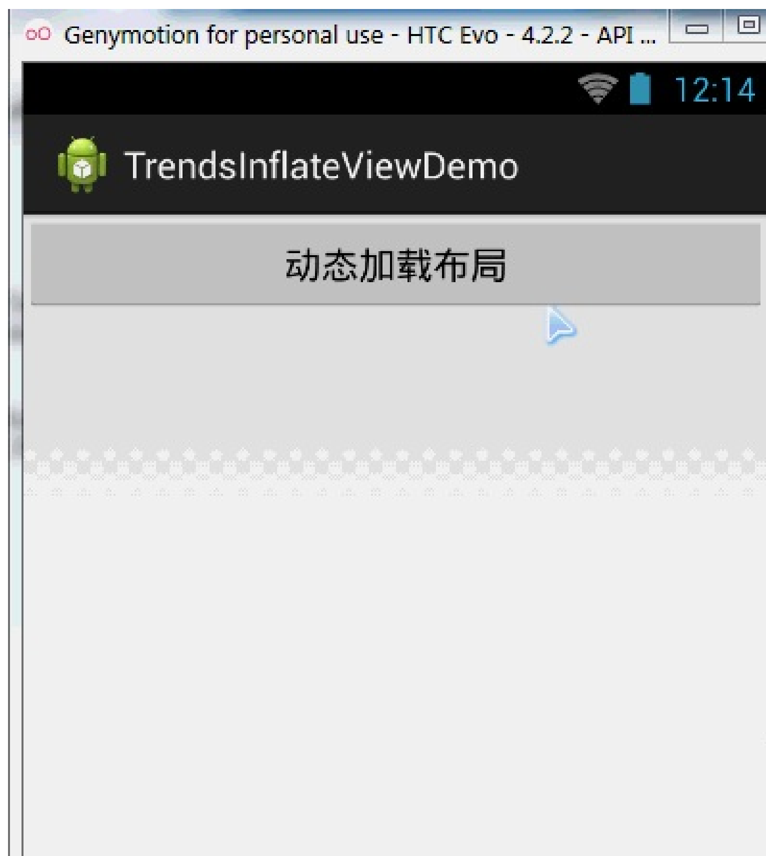
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="我是布局里的一个小按钮" />

</LinearLayout>
```

接着到我们的**MainActivity.java**在这里动态加载xml布局：

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //获得LayoutInflater对象;
        final LayoutInflater inflater = LayoutInflater.from(this);
        //获得外部容器对象
        final RelativeLayout rly = (RelativeLayout) findViewById(R.
        Button btnLoad = (Button) findViewById(R.id.btnLoad);
        btnLoad.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //加载要添加的布局对象
                LinearLayout ly = (LinearLayout) inflater.inflate(
                        R.layout.inflate_ly, null, false).findViewB
                        R.id.ly_inflate);
                //设置加载布局的大小与位置
                RelativeLayout.LayoutParams lp = new RelativeLayout
                        LayoutParams.WRAP_CONTENT, LayoutParams.WRA
                lp.addRule(RelativeLayout.CENTER_IN_PARENT);
                rly.addView(ly,lp);
            }
        });
    }
}
```

运行截图：


Genymotion for personal use - HTC Evo - 4.2.2 - API ...
12:14
TrendsInflateViewDemo
动态加载布局

代码分析：

> ①获取容器对象:
>
> ```
> final RelativeLayout rly = (RelativeLayout) findViewById(R.id.R
> ```
>
> ②获得**Inflater**对象,同时加载被添加的布局的**xml**,通过**findViewById**找到最外层的根节点
>
> ```
> final LayoutInflater inflater = LayoutInflater.from(this);
> LinearLayout ly = (LinearLayout) inflater.inflate(R.layout.infl
>    .findViewById(R.id.ly_inflate);
> ```
>
> ③为这个容器设置大小与位置信息:
>
> ```
> RelativeLayout.LayoutParams lp = new RelativeLayout.LayoutParam
>               LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CON
>       lp.addRule(RelativeLayout.CENTER_IN_PARENT);
> ```
>
> ④添加到外层容器中:
>
> ```
> rly.addView(ly,lp);
> ```

4.LayoutInflater的inflate()方法源码

> 最后提供下LayoutInflater的inflate()方法的源码吧，有兴趣的可以看看~，其实就是Pull解析而已~

```
public View inflate(XmlPullParser parser, ViewGroup root, boolean a
    synchronized (mConstructorArgs) {
        final AttributeSet attrs = Xml.asAttributeSet(parser);
        mConstructorArgs[0] = mContext;
        View result = root;
        try {
            int type;
            while ((type = parser.next()) != XmlPullParser.START_TA
                    type != XmlPullParser.END_DOCUMENT) {
            }
            if (type != XmlPullParser.START_TAG) {
                throw new InflateException(parser.getPositionDescri
                        + ": No start tag found!");
            }
            final String name = parser.getName();
            if (TAG_MERGE.equals(name)) {
```

```
                if (root == null || !attachToRoot) {
                    throw new InflateException("merge can be used o
                            + "ViewGroup root and attachToRoot=true
                }
                rInflate(parser, root, attrs);
            } else {
                View temp = createViewFromTag(name, attrs);
                ViewGroup.LayoutParams params = null;
                if (root != null) {
                    params = root.generateLayoutParams(attrs);
                    if (!attachToRoot) {
                        temp.setLayoutParams(params);
                    }
                }
                rInflate(parser, temp, attrs);
                if (root != null && attachToRoot) {
                    root.addView(temp, params);
                }
                if (root == null || !attachToRoot) {
                    result = temp;
                }
            }
        } catch (XmlPullParserException e) {
            InflateException ex = new InflateException(e.getMessage
            ex.initCause(e);
            throw ex;
        } catch (IOException e) {
            InflateException ex = new InflateException(
                    parser.getPositionDescription()
                    + ": " + e.getMessage());
            ex.initCause(e);
            throw ex;
        }
        return result;
    }
}
```

本节小结：

> 本节给大家讲解了一下Android中的LayoutInflater(布局服务)，以及动态加载View和控件 相关的东西，相信对初学控件的朋友带来帮助~好的，就说这么多，谢谢~ 

10.9 WallpaperManager(壁纸管理器)

本节引言：

本节给大家带来的是WallpaperManager(壁纸管理器)，如其名，就是手机壁纸相关的 一个API，在本节中我们会描述下WallpaperManager的基本用法，调用系统自带的 壁纸选择功能，将Activity的背景设置为壁纸背景，以及写一个定时换壁纸的例

子~ 好了，不BB，开始本节内容~ 

官方**API**文档：WallpaperManager

1.WallpaperManager的基本用法

相关方法

设置壁纸的相关方法：

- **setBitmap**(Bitmap bitmap)：将壁纸设置为bitmap所代表的位图
- **setResource**(int resid)：将壁纸设置为resid资源所代表的图片
- **setStream**(InputStream data)：将壁纸设置为data数据所代表的图片

其他方法：

- **clear**()：清除壁纸，设置回系统默认的壁纸
- **getDesiredMinimumHeight**()：最小壁纸高度
- **getDesiredMinimumWidth**()：最小壁纸宽度
- **getDrawable**()：获得当前系统壁纸，如果没有设置壁纸，则返回系统默认壁纸
- **getWallpaperInfo**()：加入当前壁纸是动态壁纸，返回动态壁纸信息
- **peekDrawable**()：获得当前系统壁纸，如果没设置壁纸的话返回null

获得**WallpaperManager**对象

```
WallpaperManager wpManager =WallpaperManager.getInstance(this);
```

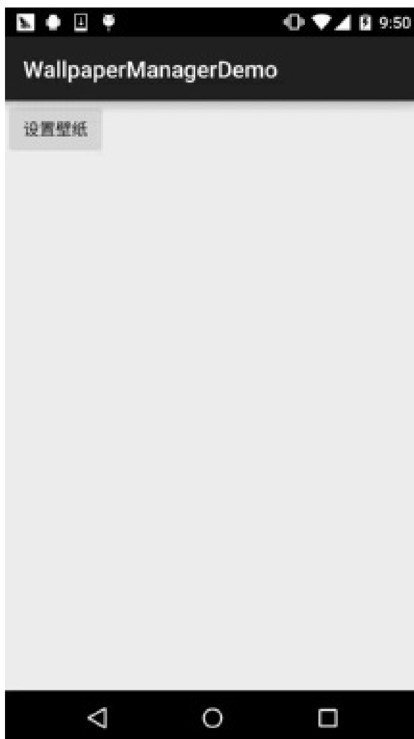
设置壁纸需要的权限

```
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
```

2.调用系统自带的壁纸选择功能

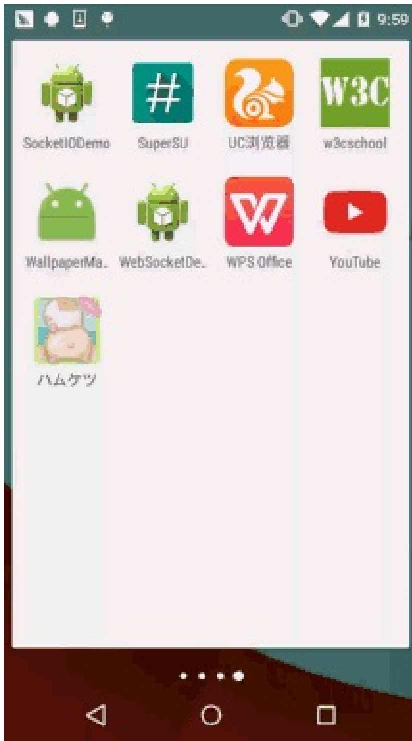
```
Button btn_set = (Button) findViewById(R.id.btn_set);
    btn_set.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent chooseIntent = new Intent(Intent.ACTION_SET_WALL
            startActivity(Intent.createChooser(chooseIntent, "选择壁
        }
    });
```

运行效果图：



3.将Activity的背景设置为壁纸背景

方法有两种，一种是在Activity中用代码进行设置，另一种是在AndroidManifest.xml中修改 Activity的主题~！



方法一：**Activity**中设置：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    setTheme(android.R.style.Theme_Wallpaper_NoTitleBar_Fullscreen)
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

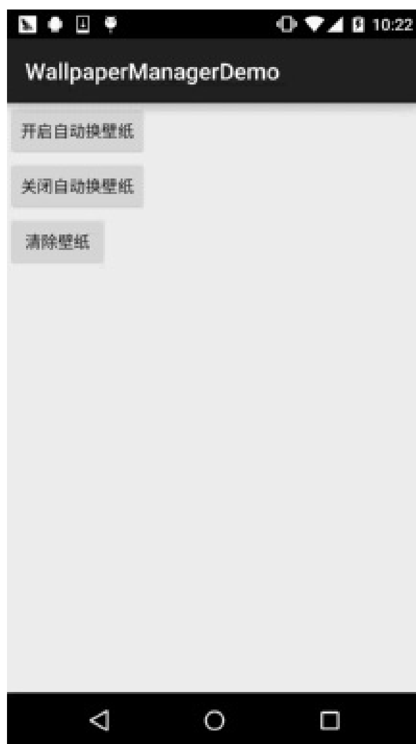
方法二：**AndroidManifest.xml**修改**theme**：

```
<activity android:name=".MainActivity"
android:theme="@android:style/Theme.Wallpaper.NoTitleBar"/>
```

4.定时换壁纸的Demo

这里用到前面学的AlarmManager(闹钟服务)，假如你对它不了解的话可以到：10.5 AlarmManager(闹钟服务)进行学习~ 下面我们来写个Demo~

运行效果图：



代码实现：

首先我们来写一个定时换壁纸的Service：**WallPaperService.java**

```
/**
 * Created by Jay on 2015/11/13 0013.
 */
public class WallPaperService extends Service {

    private int current = 0;  //当前壁纸下标
    private int[] papers = new int[]{R.mipmap.gui_1,R.mipmap.gui_2,
    private WallpaperManager wManager = null;   //定义WallpaperMana

    @Override
    public void onCreate() {
        super.onCreate();
        wManager = WallpaperManager.getInstance(this);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startIc
        if(current >= 4)current = 0;
        try{
            wManager.setResource(papers[current++]);
        }catch(Exception e){e.printStackTrace();}
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

接着撸个简单的布局，三个Button：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_on"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="开启自动换壁纸" />

    <Button
        android:id="@+id/btn_off"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="关闭自动换壁纸" />

    <Button
        android:id="@+id/btn_clean"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="清除壁纸" />

</LinearLayout>
```

接着是我们的Activity，在这里实例化aManager并设置定时事件~：**MainActivity.java**：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_on;
    private Button btn_off;
    private Button btn_clean;
    private AlarmManager aManager;
    private PendingIntent pi;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //①获得AlarmManager对象:
        aManager = (AlarmManager) getSystemService(ALARM_SERVICE);
        //②指定要启动的Service,并指明动作是Servce:
        Intent intent = new Intent(MainActivity.this, WallPaperServ
        pi = PendingIntent.getService(MainActivity.this, 0, intent,
        bindViews();
    }

    private void bindViews() {
        btn_on = (Button) findViewById(R.id.btn_on);
```

```
        btn_off = (Button) findViewById(R.id.btn_off);
        btn_clean = (Button) findViewById(R.id.btn_clean);
        btn_on.setOnClickListener(this);
        btn_off.setOnClickListener(this);
        btn_clean.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_on:
                aManager.setRepeating(AlarmManager.RTC_WAKEUP, 0, 3
                btn_on.setEnabled(false);
                btn_off.setEnabled(true);
                Toast.makeText(MainActivity.this, "自动更换壁纸设置成功
                break;
            case R.id.btn_off:
                btn_on.setEnabled(true);
                btn_off.setEnabled(false);
                aManager.cancel(pi);
                break;
            case R.id.btn_clean:
                try {
                    WallpaperManager.getInstance(getApplicationCont
                    Toast.makeText(MainActivity.this, "清除壁纸成功~"
                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
        }
    }
}
```

最后别忘了加上设置壁纸的权限以及为我们的Service进行注册：**AndroidManifest.xml**：

```
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<service android:name=".WallPaperService"/>
```

好的，非常简单~

5.本节示例代码下载

WallpaperManagerDemo.zip

本节小结：

> 好的，本节给大家介绍了下WallpaperManager的一些基本用法~更多的东西还需你们自己 进行探究~谢谢~!

10.10 传感器专题(1)——相关介绍

1.传感器相关介绍：

说到传感器，相信大家都不会陌生吧，比如微信的摇一摇就用到了加速度传感器；

传感器的定义：一种物理设备或者生物器官，能够探测、感受外界的信号，物理条件(如光，热， 适度)或化学组成（如烟雾），并将探知的信息传递给其他的设备或者器官！

传感器的种类：可以从不同的角度对传感器进行划分，转换原理(传感器工作的基本物理或化学 效应)；用途；输出信号以及制作材料和工艺等。一般是按工作原来来分：物理传感器与化学传感器 两类！手机上搭载的基本都是物理传感器，手机上搭载的传感器有下面这些：

- 方向传感器(**Orientation sensor**)
- 加速感应器（**Accelerometer sensor**）
- 陀螺仪传感器(**Gyroscope sensor**)
- 磁场传感器(**Magnetic field sensor**)
- 距离传感器(**Proximity sensor**)
- 光线传感器(**Light sensor**)
- 气压传感器(**Pressure sensor**)
- 温度传感器（**Temperature sensor**）
- 重力感应器（**Gravity sensor，Android 2.3**引入）
- 线性加速感应器（**Linear acceleration sensor，Android 2.3**引入）
- 旋转矢量传感器（**Rotation vector sensor，Android 2.3**）
- 相对湿度传感器（**Relative humidity sensor，Android 4.0**）
- 近场通信（**NFC**）传感器（**Android 2.3**引入），**NFC**和其他不一样，具有读写功能。

当然除了这些以外还有其他比如心率传感器，记步传感器，指纹传感器等， 关于Android设备所支持的传感器类型可见官方文档：Sensor Summary部分的内容~

2.如何查看自己手机所支持的传感器有哪些？

上面说的这么多种肯定不是所有手机都具备的，每台手机上搭载的传感器类型 以及数目都可能是不一样的，比如我手头上的Nexus 5支持的传感器类型有： 重力，光线，距离，气压和陀螺仪！而令意外moto x 二代则有重力，光线， 距离和红外传感器！关于自己手机支持的传感器类型，你可以到相关的评测 网站比如：中关村手机在线，太平洋等，搜索到自己的机型查看相关参数！ 当然，我们也可以自己写代码来看看自己手机支持的传感器类型~

代码示例：

运行效果图：



代码实现：

activity_main.xml：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:id="@+id/txt_show"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </ScrollView>
</RelativeLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private TextView txt_show;
    private SensorManager sm;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sm = (SensorManager) getSystemService(Context.SENSOR_SERVIC
```

```
        txt_show = (TextView) findViewById(R.id.txt_show);

        List<Sensor> allSensors = sm.getSensorList(Sensor.TYPE_ALL)
        StringBuilder sb = new StringBuilder();

        sb.append("此手机有" + allSensors.size() + "个传感器，分别有：\
        for(Sensor s:allSensors){
            switch (s.getType()){
                case Sensor.TYPE_ACCELEROMETER:
                    sb.append(s.getType() + " 加速度传感器(Accelerome
                    break;
                case Sensor.TYPE_GYROSCOPE:
                    sb.append(s.getType() + " 陀螺仪传感器(Gyroscope
                    break;
                case Sensor.TYPE_LIGHT:
                    sb.append(s.getType() + " 光线传感器(Light senso
                    break;
                case Sensor.TYPE_MAGNETIC_FIELD:
                    sb.append(s.getType() + " 磁场传感器(Magnetic fi
                    break;
                case Sensor.TYPE_ORIENTATION:
                    sb.append(s.getType() + " 方向传感器(Orientation
                    break;
                case Sensor.TYPE_PRESSURE:
                    sb.append(s.getType() + " 气压传感器(Pressure se
                    break;
                case Sensor.TYPE_PROXIMITY:
                    sb.append(s.getType() + " 距离传感器(Proximity s
                    break;
                case Sensor.TYPE_TEMPERATURE:
                    sb.append(s.getType() + " 温度传感器(Temperature
                    break;
                default:
                    sb.append(s.getType() + " 其他传感器" + "\n");
                    break;
            }
            sb.append("设备名称：" + s.getName() + "\n 设备版本：" + s
                    + s.getVendor() + "\n\n");
        }
        txt_show.setText(sb.toString());
    }
}
```

3.Sensor传感器相关的方法以及使用套路

从2中的例子我们可以大概地总结下获取Sensor传感器以及获取传感器相关的一些信息 流程如下：

1)Sensor传感器的相关方法

Step 1：获得传感器管理器：

```
SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE);
```

- **Step 2**：获得设备的传感器对象的列表：

```
List<Sensor> allSensors = sm.getSensorList(Sensor.TYPE_ALL);
```

- **Step 3**：迭代获取Sensor对象，然后调用对应方法获得传感器的相关信息：

```
for(Sensor s:allSensors){
    sensor.getName();   //获得传感器名称
    sensor.getType();     //获得传感器种类
    sensor.getVendor();    //获得传感器供应商
    sensor.getVersion();    //获得传感器版本
    sensor.getResolution();  //获得精度值
    sensor.getMaximumRange(); //获得最大范围
    sensor.getPower();        //传感器使用时的耗电量
}
```

2)传感器的使用套路

一般我们是很少说直接去获取Sensor，然后获取上面这些信息的！因为这没什么大的作用，我们更多的时候是去获取传感器采集到的数据，比如获取当前的大气压， 或者方向传感器三个角的值，或者陀螺仪的值这样~而大部分的传感器数据采集都是 下面的一个套路：

~Step 1：获得传感器管理器：

```
SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE);
```

~Step 2：调用特定方法获得需要的传感器：

比如这里获取的是方向传感器，想获得什么传感器自己查API~：

```
Sensor mSensorOrientation = sm.getDefaultSensor(Sensor.TYPE_ORIENTA
```

~Step 3：实现**SensorEventListener**接口，重写**onSensorChanged**和**onAccuracyChanged**的方法！

onSensorChanged：当传感器的值变化时会回调

onAccuracyChanged：当传感器的进度发生改变时会回调

```
@Override
public void onSensorChanged(SensorEvent event) {
    final float[] _Data = event.values;
   this.mService.onSensorChanged(_Data[0],_Data[1],_Data[2]);
}
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}
```

我们一般获取传感器数据的来源就是这个SensorEvent，这个类中有一个**values**的变量，类型是**Float[]**，该变量最多有只有三个元素，而且传感器不同，对应元素代表的含义也不同，比如方向传感器中第一个元素是方位角的值，而气压传感器中第一个值则是气压值！

~Step 4：SensorManager对象调用**registerListener**注册监听器：

```
<code>ms.registerListener(mContext, mSensorOrientation, android.ha
</code>
```

方法也很简单，对应的参数：上下文对象，**Sensor**传感器对象，以及传感器的延时时间的精度密度，有四个可选值：

- **SENSOR_DELAY_FASTEST**——延时：**0ms**
- **SENSOR_DELAY_GAME**——延时：**20ms**
- **SENSOR_DELAY_UI**——延时：**60ms**
- **SENSOR_DELAY_NORMAL**——延时：**200ms**

当然低延时意味着更频繁的检车，更意味着更多的电量消耗，如果不是要求精度非常高的建议别使用太高精度的，一般用第三个较多~自己衡量衡量吧~

~Step 5：监听器的取消注册：

用完就放，一个很好的习惯，一般我们可以把他写到Activity或者Service的销毁方法中：

```
ms.registerListener(mContext, mSensorOrientation, android.hardware
```

好的，套路非常简单~

4.本节示例代码下载：

SensorDemo1.zip

本节小结：

> 好的，本节给大家讲解了下Android中的传感器的介绍以及如何了解自己手机所支持的传感器，除了网上查，也可以自己写代码测，然后还讲解了Sensor传感器相关信息获取的方法流程，最后还讲解了采集传感器数据的套路，后面我们会针对一些常用的传感器的用法进行剖析，敬请期待~ 

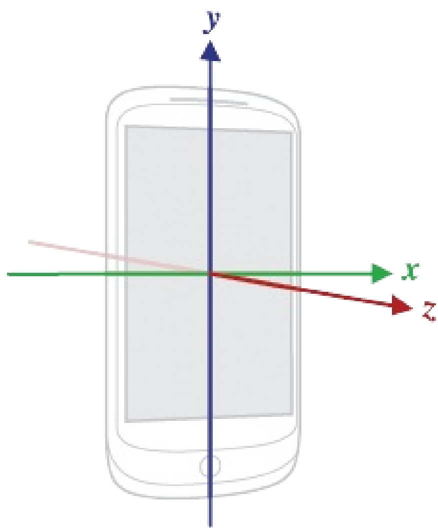
10.11 传感器专题(2)——方向传感器

本节引言：

在上一节中我们中我们对传感器的一些基本概念进行了学习，以及学习了使用传感器的套路， 本节给大家带来的传感器是方向传感器的用法，好的，开始本节内容~

1.三维坐标系的概念：

在Android平台中，传感器框架通常是使用一个标准的三维坐标系来表示一个值的。以本节 要讲的方向传感器为例子，确定一个方向也需要一个三维坐标，毕竟我们的设备不可能永远 都是水平端着的吧，安卓给我们返回的方向值就是一个长度为3的flaot数组，包含三个方向 的值！官方API文档中有这样一个图：sensors_overview



如果你看不懂图，那么写下文字解释：

- **X**轴的方向：沿着屏幕水平方向从左到右，如果手机如果不是是正方形的话，较短的边需要水平 放置，较长的边需要垂直放置。
- **Y**轴的方向：从屏幕的左下角开始沿着屏幕的的垂直方向指向屏幕的顶端
- **Z**轴的方向：当水平放置时，指向天空的方向

2.方向传感器的三个值

上一节中说了，传感器的回调方法：onSensorChanged中的参数SensorEvent event，event的 值类型是Float[]的，而且最多只有三个元素，而方向传感器则刚好有三个元素，都表示度数！ 对应的含义如下：

values[0]：方位角，手机绕着Z轴旋转的角度。0表示正北(North)，90表示正东(East)， 180表示正南(South)，270表示正西(West)。假如values[0]的值刚好是这四个值的话， 并且手机沿水平放置的话，那么当前手机的正前方就是这四个方向，可以利用这一点来 写一个指南针！

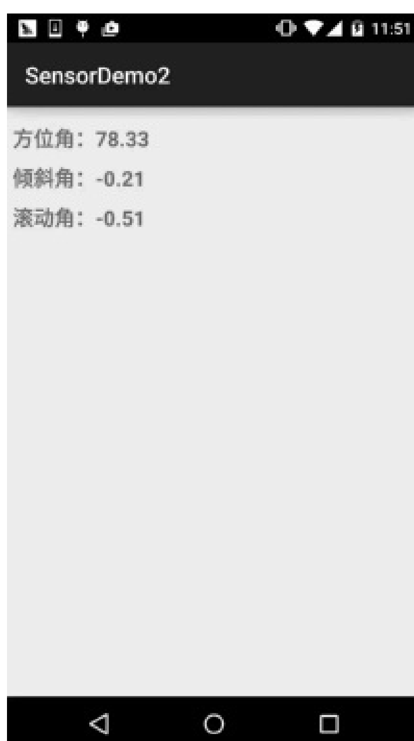
values[1]：倾斜角，手机翘起来的程度，当手机绕着x轴倾斜时该值会发生变化。取值 范围是[-180,180]之间。假如把手机放在桌面上，而桌面是完全水平的话，values1的则应该 是0，当然很少桌子是绝对水平的。从手机顶部开始抬起，直到手机沿着x轴旋转180(此时屏幕 乡下水平放在桌面上)。在这个旋转过程中，values[**1**]的值会从**0**到**-180**之间变化，即手机抬起 时，values1的值会逐渐变小，知道等于-180；而加入从手机底部开始抬起，直到手机沿着x轴 旋转180度，此时values[**1**]的值会从**0**到**180**之间变化。我们可以利用value[**1**]的这个特性结合value[**2**]来实现一个平地尺！

value[2]：滚动角，沿着Y轴的滚动角度，取值范围为：[-90,90]，假设将手机屏幕朝上水平放在 桌面上，这时如果桌面是平的，values2的值应为0。将手机从左侧逐渐抬起，values[**2**]的值将 逐渐减小，知道垂直于手机放置，此时values[**2**]的值为-90，从右侧则是0-90；加入在垂直位置 时继续向右或者向左滚动，values[**2**]的值将会继续在-90到90之间变化！

假如你不是很懂，没事我们写个demo验证下就知道了~

3.简单的Demo帮助我们理解这三个值的变化：

运行效果图：



实现代码：

布局代码：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:id="@+id/tv_value1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="方位角"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/tv_value2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="倾斜角"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/tv_value3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="滚动角"
        android:textSize="18sp"
        android:textStyle="bold" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Sens

    private TextView tv_value1;
    private TextView tv_value2;
    private TextView tv_value3;
    private SensorManager sManager;
    private Sensor mSensorOrientation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sManager = (SensorManager) getSystemService(SENSOR_SERVICE)
        mSensorOrientation = sManager.getDefaultSensor(Sensor.TYPE_
        sManager.registerListener(this, mSensorOrientation, SensorM
        bindViews();
    }

    private void bindViews() {
        tv_value1 = (TextView) findViewById(R.id.tv_value1);
        tv_value2 = (TextView) findViewById(R.id.tv_value2);
        tv_value3 = (TextView) findViewById(R.id.tv_value3);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        tv_value1.setText("方位角：" + (float) (Math.round(event.val
        tv_value2.setText("倾斜角：" + (float) (Math.round(event.val
        tv_value3.setText("滚动角：" + (float) (Math.round(event.val
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
}
```

代码非常简单~，你想真正体验下这三个值的变化，自己运行下程序转转手机就知道了~

4.一个简易版的文字指南针示例

下面我们来写个简单的文字版的指南针来体验体验，当文字显示正南的时候，表示手机 的正前方就是南方！

运行效果图：



代码实现：

自定义View：**CompassView.java**

```
/**
 * Created by Jay on 2015/11/14 0014.
 */
public class CompassView extends View implements Runnable{

    private Paint mTextPaint;
    private int sWidth,sHeight;
    private float dec = 0.0f;
    private String msg  = "正北 0°";

    public CompassView(Context context) {
        this(context, null);
    }

    public CompassView(Context context, AttributeSet attrs) {
        super(context, attrs);
        sWidth = ScreenUtil.getScreenW(context);
        sHeight = ScreenUtil.getScreenH(context);
        init();
        new Thread(this).start();
    }

    public CompassView(Context context, AttributeSet attrs, int def
        super(context, attrs, defStyleAttr);
    }

    private void init() {
```

```
        mTextPaint = new Paint();
        mTextPaint.setColor(Color.GRAY);
        mTextPaint.setTextSize(64);
        mTextPaint.setStyle(Paint.Style.FILL);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawText(msg, sWidth / 4 , sWidth / 2, mTextPaint);
    }

    // 更新指南针角度
    public void setDegree(float degree)
    {
        // 设置灵敏度
        if(Math.abs(dec - degree) >= 2 )
        {
            dec = degree;
            int range = 22;
            String degreeStr = String.valueOf(dec);

            // 指向正北
            if(dec > 360 - range && dec < 360 + range)
            {
                msg = "正北 " + degreeStr + "°";
            }

            // 指向正东
            if(dec > 90 - range && dec < 90 + range)
            {
                msg = "正东 " + degreeStr + "°";
            }

            // 指向正南
            if(dec > 180 - range && dec < 180 + range)
            {
                msg = "正南 " + degreeStr + "°";
            }

            // 指向正西
            if(dec > 270 - range && dec < 270 + range)
            {
                msg = "正西 " + degreeStr + "°";
            }

            // 指向东北
            if(dec > 45 - range && dec < 45 + range)
            {
                msg = "东北 " + degreeStr + "°";
            }
```

```
            // 指向东南
            if(dec > 135 - range && dec < 135 + range)
            {
                msg = "东南 " + degreeStr + "°";
            }

            // 指向西南
            if(dec > 225 - range && dec < 225 + range)
            {
                msg = "西南 " + degreeStr + "°";
            }

            // 指向西北
            if(dec > 315 - range && dec < 315 + range)
            {
                msg = "西北 " + degreeStr + "°";
            }
        }
    }

    @Override
    public void run() {
        while(!Thread.currentThread().isInterrupted())
        {
            try
            {
                Thread.sleep(100);
            }
            catch(InterruptedException e)
            {
                Thread.currentThread().interrupt();
            }
            postInvalidate();
        }
    }
}
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements Sens

    private CompassView cView;
    private SensorManager sManager;
    private Sensor mSensorOrientation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        cView = new CompassView(MainActivity.this);
        sManager = (SensorManager) getSystemService(SENSOR_SERVICE)
        mSensorOrientation = sManager.getDefaultSensor(Sensor.TYPE_
        sManager.registerListener(this, mSensorOrientation, SensorM
        setContentView(cView);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        cView.setDegree(event.values[0]);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        sManager.unregisterListener(this);
    }
}
```

这就是一个很简单的指南针的雏形了，有兴趣的可以自己绘制个罗盘和指针，然后实现一个 好看的指南针~

5.本节示例代码下载：

SensorDemo2.zip

SensorDemo3.zip

本节小结：

> 好的，本节给大家介绍了Android中最常用的方向传感器，以及他的简单用法，以及 写了一个指南针的例子，而完成指南针我们只用到一个values[0]的值，利用其他两个 值我们还可以用来测量某地是否平躺，即制作水平尺，有空的可以写个来玩玩~ 好的，就到这里，谢谢~ 

10.12 传感器专题(3)——加速度/陀螺仪传感器

本节引言：

本节继续来扣Android中的传感器，本节带来的是加速度传感器(Accelerometer sensor)以及 陀螺仪传感器(Gyroscope sensor)，和上一节的方向传感器一样有着x，y，z 三个轴， 还是要说一点：x，y轴的坐标要和绘图那里的x，y轴区分开来！传感器的是以左下角 为原点的！x向右，y向上！好的，带着我们的套路来学本节的传感器吧！

另外，想说一点的就是我们不是专门搞这个的，就写东西啊玩玩，见识下而已哈，很多东西 别太较真！

PS:方向传感器其实就是利用加速度传感器和磁场传感器来获取方位的，在2.2开始就被弃用了~

1.加速度传感器(Accelerometer sensor)

1）名词概念：

> - 加速度传感器的单位：加速度**(m/s^2)**
> - 方向传感器获取到的加速度是：手机运动的加速度与重力加速度**(9.81m/s^2)**的合加速度
> - 另外重力加速度是垂直向下的！

关于这个不同方向合加速度的计算好像蛮复杂的，这里我们就不去纠结这个了！ 先来看看加速度的value数组中的三个数的值吧~依旧是上节的代码，改下传感器而已~

水平放置： 竖直平放： 竖直横放：



从上面我们知道value数组的三个值分别对应X，Y，Z轴上的加速度！ 好的，知道个大概，我们来写个简易计步器来熟悉下用法！

2）.简易计步器的实现

运行效果图：



代码实现：

布局代码：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="30dp"
        android:text="简易计步器"
        android:textSize="25sp" />

    <TextView
        android:id="@+id/tv_step"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="5dp"
        android:text="0"
        android:textColor="#DE5347"
        android:textSize="100sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/btn_start"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:text="开始"
        android:textSize="25sp" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private SensorManager sManager;
    private Sensor mSensorAccelerometer;
    private TextView tv_step;
    private Button btn_start;
    private int step = 0;   //步数
    private double oriValue = 0;  //原始值
    private double lstValue = 0;  //上次的值
    private double curValue = 0;  //当前值
    private boolean motiveState = true;   //是否处于运动状态
    private boolean processState = false;   //标记当前是否已经在计步

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sManager = (SensorManager) getSystemService(SENSOR_SERVICE
        mSensorAccelerometer = sManager.getDefaultSensor(Sensor.TYP
        sManager.registerListener(this, mSensorAccelerometer, Senso
        bindViews();
    }

    private void bindViews() {

        tv_step = (TextView) findViewById(R.id.tv_step);
        btn_start = (Button) findViewById(R.id.btn_start);
        btn_start.setOnClickListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        double range = 1;   //设定一个精度范围
        float[] value = event.values;
        curValue = magnitude(value[0], value[1], value[2]);   //计算
        //向上加速的状态
        if (motiveState == true) {
            if (curValue >= lstValue) lstValue = curValue;
            else {
                //检测到一次峰值
                if (Math.abs(curValue - lstValue) > range) {
                    oriValue = curValue;
                    motiveState = false;
                }
            }
        }
        //向下加速的状态
        if (motiveState == false) {
            if (curValue <= lstValue) lstValue = curValue;
            else {
                if (Math.abs(curValue - lstValue) > range) {
                    //检测到一次峰值
                    oriValue = curValue;
                    if (processState == true) {
                        step++;  //步数 + 1
                        if (processState == true) {
                            tv_step.setText(step + "");    //读数更
                        }
                    }
                    motiveState = true;
                }
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {}
```

```
    @Override
    public void onClick(View v) {
        step = 0;
        tv_step.setText("0");
        if (processState == true) {
            btn_start.setText("开始");
            processState = false;
        } else {
            btn_start.setText("停止");
            processState = true;
        }
    }

    //向量求模
    public double magnitude(float x, float y, float z) {
        double magnitude = 0;
        magnitude = Math.sqrt(x * x + y * y + z * z);
        return magnitude;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        sManager.unregisterListener(this);
    }
}
```

好的，真的是非常简易的计步器...上面的步数是我坐着拿手撸出来的...，毕竟写来玩玩而已~

2.陀螺仪传感器(Gyroscope sensor)

1）名词概念：

陀螺仪又叫角速度传感器，一般用来检测手机姿态的，好像手机中的陀螺仪传感器一般都是三轴的！体感游戏用得最多，手机拍照防抖，GPS惯性导航，还有为APP添加一些动作感应(比如轻轻晃动手机 关闭来电铃声)等等，具体的可以自己去百度下~

- 陀螺仪传感器的单位：角速度(弧度/秒)**radians/second**
- 获得传感器用的是：**Sensor.TYPE_GYROSCOPE**

他的三个值依次是沿着X轴，Y轴，Z轴旋转的角速度，手机逆时针旋转，角速度值为正，顺时针则为负值！经常用于计算手机已经转动的角度！这是网上的一段代码~

```
private static final float NS2S = 1.0f / 1000000000.0f;
private float timestamp;

public void onSensorChanged(SensorEvent event)
{
    if (timestamp != 0)
    {
        // event.timesamp表示当前的时间，单位是纳秒（1百万分之一毫秒）
        final float dT = (event.timestamp - timestamp) * NS2S;
        angle[0] += event.values[0] * dT;
        angle[1] += event.values[1] * dT;
        angle[2] += event.values[2] * dT;
    }
    timestamp = event.timestamp;
}
```

通过陀螺仪传感器相邻两次获得数据的时间差（dT）来分别计算在这段时间内手机延X、 Y、Z轴旋转的角度，并将值分别累加到angle数组的不同元素上

3.本节示例代码下载：

SensorDemo4.zip

本节小结：

> 好的，本节给大家简单的跟大家介绍了下加速度传感器和陀螺仪，写了个简易计步器， 感觉传感器没怎么玩过，没什么好写，算了，下节就简单的把剩下的
>
> 传感器介绍下 算了，就当科普科普，以后要用到再深入研究吧~ 

10.12 传感器专题(4)——其他传感器了解

本节引言：

在上一节的结尾说了，传感器部分因为笔者没怎么玩过，本节就简单的把剩下的几个常用的 传感器介绍一遍，当作科普，以后用到再慢慢研究~

1.磁场传感器(Magnetic field sensor)

作用：该传感器主要用于读取手机附近的磁场变化

传感器的值采集到的值：有三个，分别是：X，Y，Z轴上方向上的磁场值

数值单位：T，微特斯拉

传感器获取：Sensor.**TYPE_MAGNETIC_FIELD**

2.距离传感器(Proximity sensor)

作用：用于感应手机与人体的距离，用得最多的就是手机通话时候，脸部贴近屏幕时， 屏幕会熄灭，当脸部离开屏幕一段距离后，屏幕又会亮起，这样可以避免通过过程脸部误碰 挂断按钮，从而导致通话中断~我们可以利用这个传感器来做一些交互型的App~

传感器的值采集到的值：有一个，物体与设备间的距离

数值单位：cm，厘米

传感器获取：Sensor.**TYPE_PROXIMITY**

其他：

- ①关于距离传感器可能有两种，一种是能直接给出距离的，而另一种则是给出靠近或者远离！ 就是只返回两个值，0.0或者最大值！我们可以通过对比解析度和最大值是否相等进行判断！ 假如相等说明是后者，假如不等说明是前者！
- ②调用sensor.getResolution()方法获得解析度，调用getMaximumRange()获得最大值！

3.光线传感器(Light sensor)

作用：用来读取光度值，即光线强度

传感器的值采集到的值：有一个，光亮度值

数值单位：lux，1流明每平方米面积，就是1勒克斯(lux)，最大值是：120000.0f，Android 中把光线强度分了不同的等级，可以自行查看SensorManager类~

传感器获取：Sensor.**TYPE_LIGHT**

4.气压传感器(Pressure sensor)

作用：用于测量大气压力，常用于测量海拔高度

传感器的值采集到的值：有一个，大气压值

数值单位：hPa，百帕

传感器获取：Sensor.**TYPE_PRESSURE**

5.温度传感器（Temperature sensor）

作用：测量手机内部的温度或者外部环境的问题

传感器的值采集到的值：有一个，温度值

数值单位：℃，摄氏度

传感器获取：Sensor.**TYPE_TEMPERATURE**(手机内部)/**TYPE_AMBIENT_TEMPERATURE**(手机外部)

6.传感器模拟工具——SensorSimulator

如题，当我们的真机不具备某种传感器的时候，而又需要进行开发~关于具体用法可见下面的文章： Android设备上的传感器模拟工具：SensorSimulator

本节小结：

> 好的，本节应该是基础入门系列里最鸡肋的一节了吧，本来不想写的，不过还是写下吧， 上面的东西知道下就好~还是那句话，以后要用到再深入研究~谢谢
> 

10.14 Android GPS初涉

本节引言：

说到GPS这个名词，相信大家都不陌生，GPS全球定位技术嘛，嗯，Android中定位的方式 一般有这四种：GPS定位，WIFI定准，基站定位，AGPS定位(基站+GPS)；

本系列教程只讲解GPS定位的基本使用！GPS是通过与卫星交互来获取设备当前的经纬度，准确 度较高，但也有一些缺点，最大的缺点就是：室内几乎无法使用...需要收到4颗卫星或以上 信号才能保证GPS的准确定位！但是假如你是在室外，无网络的情况，GPS还是可以用的！

本节我们就来探讨下Android中的GPS的基本用法~

1.定位相关的一些API

1）LocationManager

官方API文档：LocationManager

这玩意是系统服务来的，不能直接new，需要：

```
LocationManager lm = (LocationManager)getSystemService(Context.LOCA
```

另外用GPS定位别忘了加权限：

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCAT
```

好的，获得了LocationManager对象后，我们可以调用下面这些常用的方法：

- **addGpsStatusListener**(GpsStatus.Listener listener)：添加一个GPS状态监听器
- **addProximityAlert**(double latitude, double longitude, float radius, long expiration, PendingIntent intent)： 添加一个临界警告
- **getAllProviders**()：获取所有的LocationProvider列表
- **getBestProvider**(Criteria criteria, boolean enabledOnly)：根据指定条件返回最优LocationProvider
- **getGpsStatus**(GpsStatus status)：获取GPS状态
- **getLastKnownLocation**(String provider)：根据LocationProvider获得最近一次已知的Location

- **getProvider**(String name)：根据名称来获得LocationProvider
- **getProviders**(boolean enabledOnly)：获取所有可用的LocationProvider
- **getProviders**(Criteria criteria, boolean enabledOnly)：根据指定条件获取满足条件的所有LocationProvider
- **isProviderEnabled**(String provider)：判断指定名称的LocationProvider是否可用
- **removeGpsStatusListener**(GpsStatus.Listener listener)：删除GPS状态监听器
- **removeProximityAlert**(PendingIntent intent)：删除一个临近警告
- **requestLocationUpdates**(long minTime, float minDistance, Criteria criteria, PendingIntent intent)： 通过制定的LocationProvider周期性地获取定位信息，并通过Intent启动相应的组件
- **requestLocationUpdates**(String provider, long minTime, float minDistance, LocationListener listener)： 通过制定的LocationProvider周期性地获取定位信息，并触发listener所对应的触发器

2）LocationProvider(定位提供者)

官方API文档：LocationProvider

这比是GPS定位组件的抽象表示，调用下述方法可以获取该定位组件的相关信息！

常用的方法如下：

- **getAccuracy**()：返回LocationProvider精度
- **getName**()：返回LocationProvider名称
- **getPowerRequirement**()：获取LocationProvider的电源需求
- **hasMonetaryCost**()：返回该LocationProvider是收费还是免费的
- **meetsCriteria**(Criteria criteria)：判断LocationProvider是否满足Criteria条件
- **requiresCell**()：判断LocationProvider是否需要访问网络基站
- **requiresNetwork**()：判断LocationProvider是否需要访问网络数据
- **requiresSatellite**()：判断LocationProvider是否需要访问基于卫星的定位系统
- **supportsAltitude**()：判断LocationProvider是否支持高度信息
- **supportsBearing**()：判断LocationProvider是否支持方向信息
- **supportsSpeed**()：判断是LocationProvider否支持速度信息

3）Location(位置信息)

官方API文档：Location

位置信息的抽象类，我们可以调用下述方法获取相关的定位信息！

常用方法如下：

- float **getAccuracy**()：获得定位信息的精度
- double **getAltitude**()：获得定位信息的高度
- float **getBearing**()：获得定位信息的方向
- double **getLatitude**()：获得定位信息的纬度
- double **getLongitude**()：获得定位信息的精度

- String **getProvider**()：获得提供该定位信息的LocationProvider
- float **getSpeed**()：获得定位信息的速度
- boolean **hasAccuracy**()：判断该定位信息是否含有精度信息

4）Criteria(过滤条件)

官方API文档：Criteria

获取LocationProvider时，可以设置过滤条件，就是通过这个类来设置相关条件的~

常用方法如下：

- **setAccuracy**(int accuracy)：设置对的精度要求
- **setAltitudeRequired**(boolean altitudeRequired)：设置是否要求LocationProvider能提供高度的信息
- **setBearingRequired**(boolean bearingRequired)：设置是否要LocationProvider求能提供方向信息
- **setCostAllowed**(boolean costAllowed)：设置是否要求LocationProvider能提供方向信息
- **setPowerRequirement**(int level)：设置要求LocationProvider的耗电量
- **setSpeedRequired**(boolean speedRequired)：设置是否要求LocationProvider能提供速度信息

2.获取LocationProvider的例子

运行效果图：



由图可以看到，当前可用的LocationProvider有三个，分别是：

- **passive**：被动提供，由其他程序提供
- **gps**：通过GPS获取定位信息
- **network**：通过网络获取定位信息

实现代码：

布局文件：**activity_main.xml**：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btn_one"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="获得系统所有的LocationProvider" />

    <Button
        android:id="@+id/btn_two"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="根据条件获取LocationProvider" />

    <Button
        android:id="@+id/btn_three"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="获取指定的LocationProvider" />

    <TextView
        android:id="@+id/tv_result"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp"
        android:background="#81BB4D"
        android:padding="5dp"
        android:textColor="#FFFFFF"
        android:textSize="20sp"
        android:textStyle="bold" />

</LinearLayout>
```

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements View

    private Button btn_one;
    private Button btn_two;
```

```
    private Button btn_three;
    private TextView tv_result;
    private LocationManager lm;
    private List<String> pNames = new ArrayList<String>(); // 存放L

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lm = (LocationManager) getSystemService(Context.LOCATION_SE
        bindViews();
    }

    private void bindViews() {
        btn_one = (Button) findViewById(R.id.btn_one);
        btn_two = (Button) findViewById(R.id.btn_two);
        btn_three = (Button) findViewById(R.id.btn_three);
        tv_result = (TextView) findViewById(R.id.tv_result);

        btn_one.setOnClickListener(this);
        btn_two.setOnClickListener(this);
        btn_three.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_one:
                pNames.clear();
                pNames = lm.getAllProviders();
                tv_result.setText(getProvider());
                break;
            case R.id.btn_two:
                pNames.clear();
                Criteria criteria = new Criteria();
                criteria.setCostAllowed(false);   //免费
                criteria.setAltitudeRequired(true);  //能够提供高度信
                criteria.setBearingRequired(true);   //能够提供方向信
                pNames = lm.getProviders(criteria, true);
                tv_result.setText(getProvider());
                break;
            case R.id.btn_three:
                pNames.clear();
                pNames.add(lm.getProvider(LocationManager.GPS_PROVI
                tv_result.setText(getProvider());
                break;
        }
    }

    //遍历数组返回字符串的方法
    private String getProvider(){
        StringBuilder sb = new StringBuilder();
        for (String s : pNames) {
```

```
            sb.append(s + "\n");
        }
        return sb.toString();
    }
}
```

3.判断GPS是否打开以及打开GPS的两种方式

在我们使用GPS定位前的第一件事应该是去判断GPS是否已经打开或可用，没打开的话我们需要去 打开GPS才能完成定位！这里不考虑AGPS的情况~

1）判断GPS是否可用

```
private boolean isGpsAble(LocationManager lm){
    return lm.isProviderEnabled(android.location.LocationManager.GP
}
```

2）检测到GPS未打开，打开GPS

方法一：强制打开GPS，Android 5.0后无用....

```
//强制帮用户打开GPS 5.0以前可用
private void openGPS(Context context){
    Intent gpsIntent = new Intent();
    gpsIntent.setClassName("com.android.settings", "com.android.set
    gpsIntent.addCategory("android.intent.category.ALTERNATIVE");
    gpsIntent.setData(Uri.parse("custom:3"));
    try {
        PendingIntent.getBroadcast(LocationActivity.this, 0, gpsInt
    } catch (PendingIntent.CanceledException e) {
        e.printStackTrace();
    }
}
```

方法二：打开GPS位置信息设置页面，让用户自行打开

```
//打开位置信息设置页面让用户自己设置
private void openGPS2(){
    Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SET
    startActivityForResult(intent,0);
}
```

4.动态获取位置信息

这个非常简单，调用requestLocationUpdates方法设置一个LocationListener定时检测位置而已！

示例代码如下：

布局:**activity_location.xml**：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_show"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="5dp"
        android:textSize="20sp"
        android:textStyle="bold" />

</LinearLayout>
```

LocationActivity.java：

```
/**
 * Created by Jay on 2015/11/20 0020.
 */
public class LocationActivity extends AppCompatActivity {

    private LocationManager lm;
    private TextView tv_show;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_location);
        tv_show = (TextView) findViewById(R.id.tv_show);
        lm = (LocationManager) getSystemService(Context.LOCATION_SE
        if (!isGpsAble(lm)) {
            Toast.makeText(LocationActivity.this, "请打开GPS~", Toas
            openGPS2();
        }
        //从GPS获取最近的定位信息
        Location lc = lm.getLastKnownLocation(LocationManager.GPS_P
        updateShow(lc);
        //设置间隔两秒获得一次GPS定位信息
```

```
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 20(
            @Override
            public void onLocationChanged(Location location) {
                // 当GPS定位信息发生改变时，更新定位
                updateShow(location);
            }

            @Override
            public void onStatusChanged(String provider, int status

            }

            @Override
            public void onProviderEnabled(String provider) {
                // 当GPS LocationProvider可用时，更新定位
                updateShow(lm.getLastKnownLocation(provider));
            }

            @Override
            public void onProviderDisabled(String provider) {
                updateShow(null);
            }
        });
    }

    //定义一个更新显示的方法
    private void updateShow(Location location) {
        if (location != null) {
            StringBuilder sb = new StringBuilder();
            sb.append("当前的位置信息：\n");
            sb.append("精度：" + location.getLongitude() + "\n");
            sb.append("纬度：" + location.getLatitude() + "\n");
            sb.append("高度：" + location.getAltitude() + "\n");
            sb.append("速度：" + location.getSpeed() + "\n");
            sb.append("方向：" + location.getBearing() + "\n");
            sb.append("定位精度：" + location.getAccuracy() + "\n");
            tv_show.setText(sb.toString());
        } else tv_show.setText("");
    }

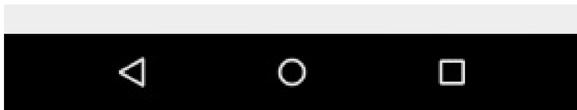
    private boolean isGpsAble(LocationManager lm) {
        return lm.isProviderEnabled(android.location.LocationManage
    }

    //打开设置页面让用户自己设置
    private void openGPS2() {
        Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_
        startActivityForResult(intent, 0);
    }

}
```

好的，非常简单，因为gps需要在室外才能用，于是趁着这个机会小跑出去便利店买了杯奶茶，顺道截下图~


GPSDemo
当前的位置信息：
精度：113.56855998
纬度：22.37494458
高度：29.0
速度：0.0
方向：0.0
定位精度：8.0
GPSDemo
当前的位置信息：
精度：113.56902416
纬度：22.37528265
高度：8.0
速度：1.0
方向：30.1
定位精度：4.0
GPSDemo
当前的位置信息：
精度：113.56958831
纬度：22.37483296
高度：19.0
速度：0.25
方向：346.3
定位精度：45.0



requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)

当时间超过minTime（单位：毫秒），或者位置移动超过minDistance（单位：米），就会调用listener中的方法更新GPS信息，建议这个minTime不小于60000，即1分钟，这样会更加高效而且省电，加入你需要尽可能 实时地更新GPS，可以将minTime和minDistance设置为0

对了，别忘了，你还需要一枚权限：

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCA
```

5.临近警告(地理围栏)

嗯，就是固定一个点，当手机与该点的距离少于指定范围时，可以触发对应的处理！ 有点像地理围栏...我们可以调用LocationManager的addProximityAlert方法添加临近警告！ 完整方法如下：

addProximityAlert(double latitude,double longitude,float radius,long expiration,PendingIntent intent)

属性说明：

- **latitude**：指定固定点的经度
- **longitude**：指定固定点的纬度
- **radius**：指定半径长度
- **expiration**：指定经过多少毫秒后该临近警告就会过期失效，-1表示永不过期
- **intent**：该参数指定临近该固定点时触发该intent对应的组件

示例代码如下：

ProximityActivity.java：

```
/**
 * Created by Jay on 2015/11/21 0021.
 */
public class ProximityActivity extends AppCompatActivity {
    private LocationManager lm;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_proximity);
        lm = (LocationManager) getSystemService(Context.LOCATION_SE
        //定义固定点的经纬度
        double longitude = 113.56843;
        double latitude = 22.374937;
        float radius = 10;     //定义半径，米
        Intent intent = new Intent(this, ProximityReceiver.class);
        PendingIntent pi = PendingIntent.getBroadcast(this, -1, int
        lm.addProximityAlert(latitude, longitude, radius, -1, pi);
    }
}
```

还需要注册一个广播接收者：**ProximityReceiver.java**：

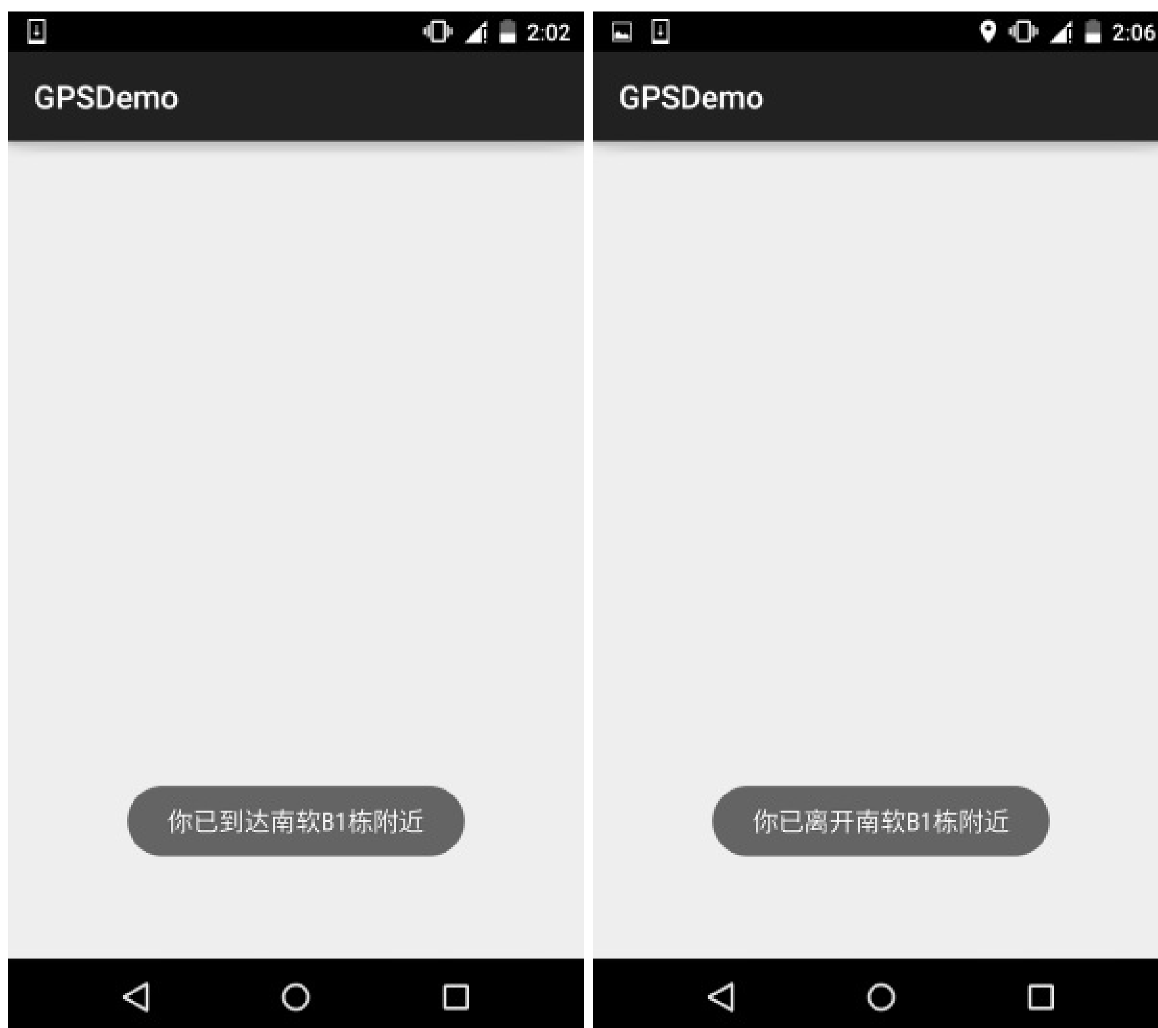
```
/**
 * Created by Jay on 2015/11/21 0021.
 */
public class ProximityReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        boolean isEnter = intent.getBooleanExtra( LocationManager.K
        if(isEnter) Toast.makeText(context, "你已到达南软B1栋附近", To
        else Toast.makeText(context, "你已离开南软B1栋附近", Toast.LEN
    }
}
```

别忘了注册：

```
<receiver android:name=".ProximityReceiver"/>
```

运行效果图：

PS：好吧，设置了10m，结果我从B1走到D1那边，不止10m了吧...还刚好下雨



6.本节示例代码下载

GPSDemo.zip

本节小结：

> 好的，本节给大家介绍了Android中GPS定位的一些基本用法，非常简单，内容部分参考的 李刚老师的《Android疯狂讲义》，只是对例子进行了一些修改
>
> 以及进行了可用性的测试！ 本节就到这里，谢谢~

11.0《2015最新Android基础入门教程》完结散花~

引言：

从六月底就开始编写这套教程，历时将近五个多月，今天终于写完了，全套教程正文部分148篇， 十大章，从基本UI控件到四大组件，Intent，Fragment，事件处理，数据存储，网络编程，绘图与动画， 多媒体，系统服务等都进行了详细的讲解！代码都是都是在Android Studio上进行编写的，全文 采用Markdown，行文结构清晰，还结合了实际开发中一些常见的问题进行了剖析...由于个人能力的局限，虽然竭尽全力，但是难免还有有一些错误纰漏，望读者海涵指出，万分感激！在写这套教材的过程中， 感触良多，借着完结散花这最后一节一吐而快，也算是暂时告别自己博客生涯的一笔吧...  一吐而快~

1.此套教程的由来

记得那是在五月份的某一天晚上，刚和舍友打完撸啊撸，玩起手机来。不经意的我加了w3c鸟巢的 公众号，然后看了下推送过来的文章，感觉有点意思，于是乎就到度娘上搜了下"w3c鸟巢"。发现有 个菜鸟教程的栏目，然后里面的教程大部分都是Web类的基础教程，而我看到了移动端的教程，上面 有着"学习Android"！这样一个教程 

，作为一个搞Android的，按照故事情节，我肯定会马上 去点开这个链接，然后发生点什么事吧...然而，我并没有点...所以故事到这里就结束了，哈哈... 当然，最后还是点了，不过在这个期间和舍友下去喝了碗糖水而已~因为年代久远，但是教程里的内 容我都已忘记，但我现在还记得，在我的柜子底有一本《Android疯狂讲义》，大学买的第一本 编程书，哈哈，可惜看了100来页我已经放弃了，一本中文版的API文档哈...也就是因为这本书，才 会小猪Android入门之路的专栏，当时抱着试一试的心情，加了w3c大师姐的微信，然后问她需不需 要一个写Android基础教程的，接着把小猪入门之路的链接发给他了，然后大师姐貌似非常的高兴， 然后又问卖不卖版权之类的， 卖版权  ？那不是有钱收咩？ 作为一个苦逼学生狗，写点东西有钱收，想想还有点小激动呢，结果兴奋了一晚上，脑子里想了 很多...后来也不知道自己是怎么想的，就跟FK(w3c鸟巢的站长)聊了下， 然后就决定在w3c鸟巢的菜鸟教程上写一套Android的基础教程，免费，嗯，没错，不收一分钱， 前提是教程不用于商业用途，原因可能是被FK的分享的精神所渲染吧，也可能是自己真的想去写 一套Android教程吧，大部分大牛没时间或者不屑于去写基础入门教程，那么就让我这个渣渣来写 吧！为后面的初学者铺铺路也好嘛~

接下来就是用百度脑图来构思入门系列要讲解的内容，学了下Markdown语法，然后就开搞，一开始 是不想在coder-pig上写的，毕竟上面有很多太监了的教程。然后开了个小号，打算在上面写这套 新教程，但是访问量却惨不忍睹，假如你是一个写博客的，看到自己花了很多时间写出来的东西， 却没人看的时候，心里肯定不舒服是吧...后来还是默默地搬回了coder-pig上，然后把第一章写 完，也开始在w3c鸟巢上发布了！接着每天就开始下面这种一成不变的枯燥的生活： 每天上班，一有时间就构思今天写什么知识点，写个什么样有趣的例子，然后晚上5点半下班后， 去吃个饭，然后就回来埋头苦写，每天晚上基本上都是我锁门的，一般十点半左右走吧，记得 最晚一次写得太嗨没注意时间写到12点半，写完看了下时间，卧槽，十二点半！！！

吓得我赶忙收拾东西，拔腿就跑，因为园区这边好像是12点就关后门的，一到那里发现门是关 着的，心理顿时凉了一大半，妈蛋，难道今晚真的要睡公司么...后来走进一看才发现门是虚掩的， 并没锁，最后还是顺利地回到了宿舍...周六日一般也没什么节目，都是回公司码字，偶尔 天气好就去跟别人打打羽毛球，大部分时间还是花在码字上，就这样坚持了五个多月，这套教程 也总算完结了~ 

此刻的心情，有点小高兴，也有点小激动，但更多的却是一种解脱，总算写完了~~~ 为何是解脱，不急，还请听我娓娓道来...

2.扒一扒我的一些情况

不用到群里问猪神在那里高就，月薪多少，做我徒弟之类的话了，现在就扒一扒自己的一些情况吧！ 今年的应届毕业生(15届)，学校是北京理工大学珠海学院(北理珠)，目前在南方软件园这边工作，是 一枚**Android**实习生，月薪也只有**3K**，五险一金什么鬼都没有！嗯，你没看错，我是一名 **3K**实习生，或许你会觉得我在开玩笑，但这就是事实，因为自己大学时候的任性，我现在还有 两门科目没有过：高数上和下，所以还没拿到毕业证...很失望是吧，还以为写这套教材的是哪位 大牛，结果是一个实习生么，哈哈~

嗯，说下自己的当前的Android技术水平吧：

中下，或者说中下也达不到，可以单独完成小型的项目！但是架构什么的，屎一比，根本不考虑 复用之类的，可以说是任意拼凑起来的垃圾，很多新兴起的技术，听过但是没有花时间去研究...

接着说说自己的工作经历吧：

2015.2

学校春节招聘会，找的第一份实习，在拱北跨境工业区那边，一家外包公司，说是公司 还不如说是工作室，加起来就那么7个人，后来还跑了个HR。在这个公司呆了一个来月，收获就是： 学会了去看官方的API文档，而非啃李刚；学会了改Hosts；知道了Fragment的用法； 写了华仔天地(刘德华粉丝俱乐部APP)的UI；各种打杂； 他们有一套自己的东西，其实就是将一些常用到的功能丢到一个Jar包里，比如图像异步加载， 图片大小的动态计算等...要什么功能问后面的，没错，没文档...所有的APP都是那个套路， 可能外包公司都是这样吧，只在乎结果而不在乎过程，另外最让我接受不了的是测试， 叫我和美工在那里划屏，只要程序不crash就好了，这就叫测试...



于是乎，我离开了这家公司，此时我实习2.8k，转正3.2k！

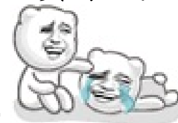
2015.4

接着我又找了另一份工作，在清华科技园那边，这就不是一家外包公司了，他们主要是 做安防和智能家居类的，氛围还是不错的，偶尔会有技术问题的撕比，周五下午还有技术交流， 而且给我配了个新的电脑和显示器，感觉自己在这里呆肯定会很嗨皮！第一个月看看文档什么的， 看看要接手的项目什么的，小日子还是过得挺滋润的，可是好景不长，做了3年的那个老员工要走了！ 他手上的两个项目都丢给了我，而且我还要开始搞另一个新的项目，这没什么，勉勉强强还可以 扛下来，但是那两个接手的项目有个要改，而且要出版本，我连代码都还没来得及熟悉...怎么玩得过来啊，自己做不过来，又不想耗时间，到时期限到了我什么都没搞出来，这样还拖累了别人! 记得想辞职前的那周过得非常的压抑，机缘巧合，好像是周三的下午吧，收到了现在公司HR打来 的电话，然后和现在的经理电话面试了下，问了一些Android基础的东西，聊得还蛮嗨的，然后约 个时间见见面，然后周五就过来面试了，再接着聊了一下现在公司的一些情况，第一感觉公司 环境还可以吧，位置都比较宽敞，然后跟他说了下我还没拿到毕业证的事，能不能转正， 可能是他当时口爽，说没什么跟人事那边说下就好...然而我在这里蹲半年了，还是实习... 然后周一回公司提交了辞职申请，然后离开了第二家公司，在这个公司的一个月，扩展了一下 自己的视野，知道了NDK和视频编解码这些东西~此时我试用3.8k，转正4.2k！

2015.5

嗯，辞去第二间公司的工作后，在学校嗨了一个星期，随手把自己的毕设给弄 完了，被迫分割成两个应用的毕设：海绵表表和一起啪啪啪，现在看来那两个自己写出来的东西， 无法直视，后来把毕设卖了，200块... 然后周一的时候就来到了现在的这家公司，又是接手项目，原来这里的那个Android开发的大牛 要跳到魅族去，第一次感觉到大牛的气息，假如他并没有走而是继续呆着多好呢？ 或许我此时又会是另外一种不同的结局了是吧~从SVN过渡到Git，从图形化界面过渡到命令行； 知道了注解，RxJava，okhttp，github，多渠道打包等等，愉快地相处了一周后，大牛走了， 接下来就是我自己看项目了，感觉就像来到一个新大陆一样，很多东西我以前都没见过， 就这样嗨了将近一个月，公司招到了另一名Android开发的，一开始听说是三年工作经验， 感觉有人带我飞了，然而事与愿违，在他身上我并没有感觉到一股大牛的气息，感觉可能是在 这个行业呆了三年吧，水平很一般，和自己比

的话可能业务经验多一点吧，跟他讨论md他听都 没听过，Android Studio也不知道，其他的更不用说，记得有一次问他一个简单的控件怎么 自定义，他的回答是：网上找下改改就能用，我想问的是实现的思路，得到回答是：知道怎么用就好... 嗯，好吧！三年嘛，项目肯定是他来接手的啦，而经理丢给了我另外一个项目， 一个无人机上绑手机测量基站天线角度等信息，然后通过wifi显示到地面上的另一台手机上， 手机自身数据采集和数据传输到没什么，难点是串口通信(FTDI)的东西，手机 通过OTG线连单片机，完成指令收发，看着API文档撸了一个星期，连个最简单的Demo都写不出来 有发没收...同样的情况又持续了一个星期，好吧，写不出东西的感觉真的很不爽，后来没办法， 只能反编译别人的apk了，花了两天时间把别人apk里的代码抽取出最关键的部分，从6000多行 的代码变成500多行的代码，看到单片机上的收发信号灯闪烁，还蛮有成就感的！ 可惜好景不长经理说要加个实时视频播放的，我真是....这玩意我都没搞过，怎么玩，于是到 Github上找了，几个开源的视频直播项目，后来还是找了WifiCarema作为研究项目，然后因为 h264库编译的问题纠结了差不多两个月，结果还是没有解决，结果项目外包给了北京那边的人 做，嗯，我的第一个项目就这样阉割了...接着做了一个很简单的小东西，再接着就到现在 就是一直在跟踪解决websocket的问题了~我司推送并不是用的第三方，而是自己用socketio 搭建的一个推送平台，用socketio的原因是三个平台都可以用一套嘛，iOS，Android，还有 web端，然后出现了漏掉报文或者收不到位置更新的问题，到现在还没找到问题发生的原因， 连问题都重现不了，我们这边一直测都没问题，一到客户手里就各种问题... 现在还在纠结这个问题中...来这里半年了，还是实习生，实习工资3k，毕业证起码要明年六月份才能 拿到，应该没得转正了，唉..

嗯，上面就是我今年到现在的一些情况，前段时间去追梦网络面试，和面试官谈了谈自己 当前的一些情况，他说感觉我走了野路子，很多东西都走偏了，然后跟我说毕业这一年 很关键，以后成型了就难改了，然后又谈了一些架构的东西，嗯，第一次那么想进一家公司， 哪怕实习两个月也好，嗯，很遗憾，结果并没有拿到offer，不过也很感谢全齐大神给自己 上了一堂课，总算知道自己接下来要去学点什么~然后又面了两家，没什么感觉，不是自己 向往的类型，最后投了一波魅族实习生，哈哈，连面试的机会都没有，这是第一次，估计 HR连简历都没看到吧~ 
以上就是我的一些个人情况的描述了，我真的是只有**3K**的实习狗，所以群里各位10k的 大老爷们，别逢年过节就叫我这个穷比发红包了... 

3.一些自学心得以及资源分享

> 怎么学Android，这可能是初学者问得最多的问题了，通过上面你也知道了小猪有多屎， 所以下面这些都是鄙人对于自学的一些浅显的看法而已，不喜请喷~

1）看书

入门推荐的书：

- 《第一行代码》：这本就不用说啦，郭霖大神写的书，入门必备
- 《**Android**群英传》：这本是医生(徐宜生)写的，嘿嘿，双11买的，今天刚收

到， 翻了下，感觉内容还是蛮简单，适合看完第一本书，或者会点Android的~

可能有的朋友会说，还有李刚Android疯狂讲义咧...嗯，买来当字典查也可以， 但是感觉看上面两本会让你更快入门，另外，在看第一行代码的时候，你也可以配合着小猪写的基础入门教程一同服用，效果更佳~

进阶推荐的书：

也是接下来自己想入手的几本书：

- 《**Android**源码设计模式解析与实战》：何红辉(Simple哥)，关爱民(爱哥)两人的大作，既可以学 习到设计模式，也可以体会到Android中蕴含的一些设计思想！
- 《**Android**开发艺术探索》：任玉刚，侧重于Android知识的体系化和系统工作机制的分析
- 《深入解析**Android 5.0**系统 》：剖析了最新Android 5.0 系统主要框架的原理和具体实现~

上述几本书我都还没摸过(还没入手)，都是广受好评的几本书~这里也推荐下!

2）看视频

网上关于Android的视频教程有很多，这里分享下基神力荐的黑马教程吧：

黑马28期Android全套视频无加密完整版：密码：h7jz

52期不加密版：密码：zve8

当然下面这些视频学习网站也很不错，也推荐下

慕课网

极客学院

麦子学院

3）看别人的技术博客

- CodeKK —— 专注于开源项目源码解析及优秀开源项目的分享
- Trinea —— 性能优化、源码解析
- 老罗的Android之旅 —— Android系统源代码分析
- 开发技术前线 —— 《Android源码设计模式》作者 Mr.Simple 维护的社区网站
- 爱哥 —— 《Android源码设计模式》作者 关爱民
- 任玉刚 —— 《Android开发艺术探索》作者 CSDN博客
- 郭霖 —— 《第一行代码》作者 CSDN博客
- 鸿洋 —— CSDN 博客专家
- 胡凯 —— 专注性能优化
- 张明云 —— Android学习之路
- Drakeet —— 贝壳单词APP开发者

- 徐宜生 —— 《Android群英传》作者
- 代码家 —— 著名博主
- 脉脉不得语 —— 著名博主
- 高建武 —— 专注性能优化，简书著名博主
- 程序亦非猿—— 简书著名博主
- 廖祜秋liaohuqiu_秋百万 —— 淘宝职员
- hi大头鬼hi —— 对RxJava有较深的研究
- 阳春面 —— 简书著名博主
- 夏安明 —— CSDN 博客专家
- 兰亭风雨 —— CSDN 博客专家
- 赵凯强 —— CSDN 博客专家
- qinjuning —— CSDN 博客专家
- 工匠若水 —— CSDN 博客专家
- 张兴业 —— CSDN 博客专家
- Coder-pig —— CSDN 博客专家，最佳入门专栏
- Keegan小刚 —— 分享了多篇Android样式的文章
- 郑海波 —— CSDN博主，文章大多与自定义控件相关
- 吴小龙同学 —— 分享了多篇关于AndroidDesignSupportLibrary的文章
- 全速前行 —— CSDN 博客专家，主讲实战技巧和平常遇到的问题

4）高质量Android社区

> - Stackoverflow —— 国外著名的问答社区
> - V2ex —— V2ex Android板块
> - Android 开发技术周报 —— 长期更新最新前言资讯
> - 开发技术前线 —— 《Android源码设计模式》作者 Mr.Simple 维护的社区网站
> - 泡在网上的日子 —— 大量第三方控件基地
> - 开源中国 —— OsChina
> - 23code —— android经典开源代码分享
> - App开发者 —— 分享Android/IOS/Swift开发和互联网内容
> - JavaApk.com —— 安卓demo聚集地，部分源码需购买VIP
> - DevStore —— 各种Demo，以及第三方服务

5）官方学习网站/Wiki

> - Android Developer
> - Android Developer(无需梯子)
> - Android Training 中文版
> - Material Design 中文版
> - Android Weekly 中文版
> - 极客学院 Wiki

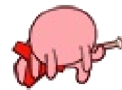
6）代码/项目下载

> 嗯，大部分时间我都会选择到Github上面找，有很多开源的第三方，下面这个务必Star：
>
> Android 开源项目分类汇总
>
> 然后笔者也分享下以前在某宝花了50多块买的一些代码吧：
>
> 5000套Android源码 密码：6we6 3175套iOS源码 密码：53v9
>
> 上面的这套代码很多都是重复的，而且大部分都是基于Eclipse，涵括的还是比较广的，可以一下！

7）梯子工具

> 嗯，假如你不想经常改hosts或者不想买vpn，但是想用Google的话，那么你可以使用蓝灯(Lantern)~ 自己搜"Lantern"下载吧~

8）一些其他的碎碎念：

嗯，上面的资源大部分来自于：Android学习资源网站大全，请务必Star！！！后续如果 有什么新的资源都会在上面进行更新，也欢迎大家share自己的一些收藏，上面的内容是小猪 群里的第一大手——基神所写，当然还有B神和曹神，街神等，这里非常感谢各位一直以来对我的 一些指导以及帮助~ 

不知道你看到上面的资源是不是，收藏收藏，买买买，下下下~

我想说的是，收藏了不去看，只是一个Url而已；下载了不去看，只是一堆数据而已； 买了书不去看，也只是一沓纸！不要让你自己只是看起来很忙很努力的样子，装比给谁看？学到手的东西才是自己的，很喜欢这样一句话："技术之路最公平也最残酷的原因是：没有捷径，需要日积月累的积累，以及对技术持久的热情。" 还记得很久之前看的锤子科技的射角设计总监罗子雄仔tedx上演讲的："如何成为一名优秀的设计师"说过的这么一段话： 格拉德威尔在《异类》一书中指出："人们眼中的天才，并非卓越非凡，而是付出了持续不断的努力， 一万小时的锤炼是任何人从平凡变成超凡的必要条件。"一万小时，也就是说你每天工作**8**小时的时间， 每周工作**5**天，你需要**5**年。你无需天才，无需智商过人，无需三头六臂，无需头上长角，你只需要持 续的、坚持的努力，有正确的方法，就能够在设计领域，一个专业中独当一面。 尽管他讲的是设计，但是很多东西都是相通的，嘿嘿，无情地上了一大碗鸡汤~ 总结下自学，无非： 多看书，看博客，做项目，看源码，不断的总结反思，让自己所学的东西所学的东西结构化！

4）一些答疑

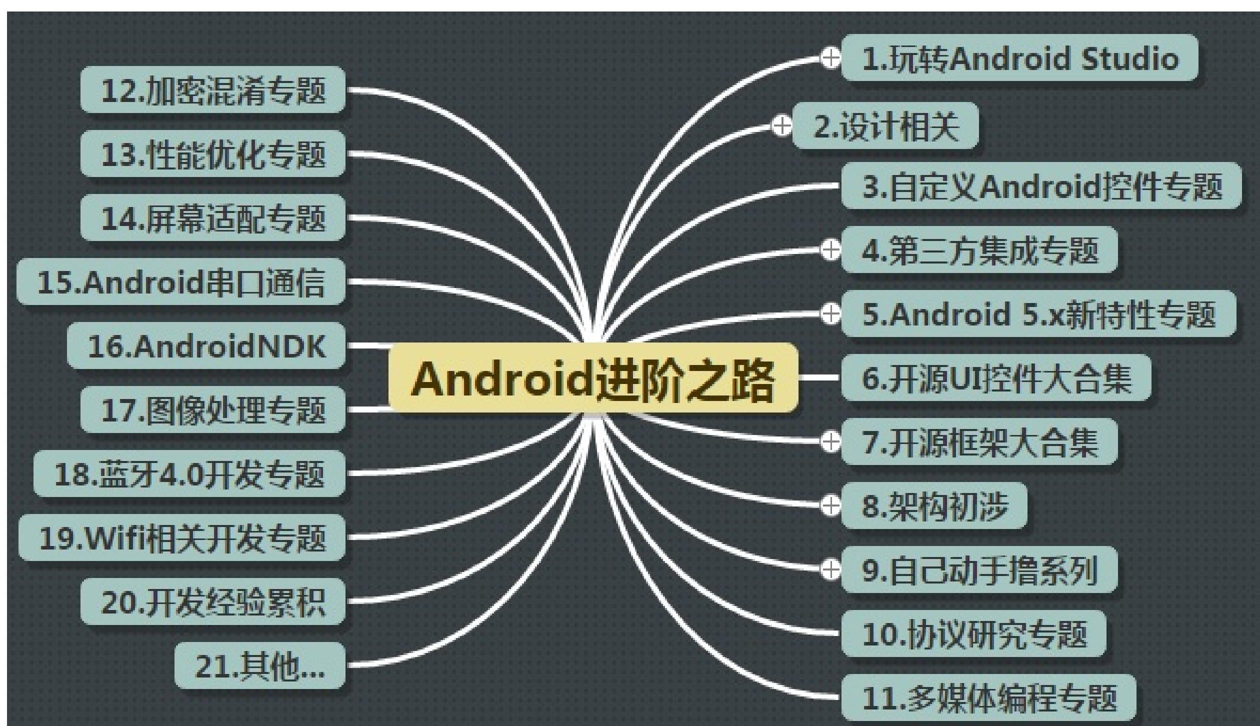
下面是一些读者经常问到的问题，下面统一答复下：

1.我是以前学**XX**的或者我不是搞编程的，我想来学**Android**，能学好不？之类的问题！ 答：前段时间在医生(徐宜生)的新浪微博看到，一位65岁的大爷，到他的公司向他请问 Android Studio，看到这里，你觉得上面的问题是问题吗？

2.XXX报错了？怎么办之类的问题 答：这种最频繁，其实很多都可以在度娘或者谷哥上找到答案，这么多人搞Android难道就 你一个人出现过这样的问题吗？或者到Stackoverflow上提问等，先自己搜过思考过，再去 问别人！！！而且别人也没有回答你的义务，别搞得好像别人不回答你的问题就很什么， 然后就恶言相向！注意问问题的技巧，整理语言，发log，出错位置代码等！

3.想加小猪做好友，为什么我拒绝了？ 答：不知道你在哪看到了我的QQ，然后看了我写的东西，就迫切的想加我为好友， 我想问，加了，然后呢？问问题更方便了？刚开始加我的我都会家，一般都是问问题， 我每次都会很耐心的解答，然后就开始依赖我了，一出问题就找我...一个两个没什么， 慢慢地人越来越多，我每天的时间都基本用在回答问题上了，结果自己一天下来什么都没做成... 不是说小猪高冷或者看不起初学者之类的，我也有自己的事要做，希望各位可以体谅下！ 有问题，可以到群里问，管理们都是很热心的，当然，前提是你的问题别一百度就可以 找到的...别做伸手党！！！

4.基础入门教程写完，那么什么时候开始写进阶教程？ 答：大家对基础入门教程的反馈都觉得写的不错，也受到了很多的好评，表扬，很感谢~ 至于进阶教程，在写基础入门的过程中就曾经简单的构思过，用百度脑图列了下大纲：



当时想着写完入门休息一个月，然后就开始写进阶系列的，大概一个月一个专题这样。 不过，进阶部分可能不会继续写，可能你会不解，为什么不写呢？坦白说下自己的一些难处吧：

首先是：花在写教程上的时间，一篇简单的教程至少需要花费我2个多小时的时间，尽管内容 比较简单，而复杂一点的，我可能需要花上2，3天！写教程不同于写笔记，要描述清晰， 写例子，贴运行效果等，笔记自己看懂就好，而教程你要让别人也看懂...

接着是：自己的进步缓慢，写完这套基本教程，和写之前的我相比，并没有什么进步； 依旧还是以前的水平...每次去面试来来去去都是说那几个破旧的项目，一点意思都没有， 我想花点时间做点什么~想学的东西有太多太多，比如，从5月份我就开始接触rxjava，然后 现在烂大街了，我还只会最简单的玩法~ 最后是：写教程不会给我带来任何的收入，上面也说了，我是一个3K实习狗，而写这套 教程是没有任何收入的，而且每个月偶尔还要给几块钱给七牛，因为图都是用的七牛的 图床，万恶的爬虫网站，把我的文章都爬过去了，然后还不注明出处，然后拼命下我的图... 这是10月份到11月份的下载流量！ 下载流量 23.49 GB

我不是富二代，记得之前也说过，我爸得了抑郁症，没工作能力的事，我妈在老家陪我爸， 也就是没收入来源，所幸的是我爸没事熬过来了，而且我已经不用每年再去支付2W的学费；尽管每月3k的工资可以维持我的生活，但是作为家里的长子，总得扛起家里的大梁吧！ 毕竟还有在读大学的弟弟和妹妹，假如我能有毕业证，现在的情况可能好一点吧！ 算了，过去的事就过去了，更重要的是以后！我也想每天闲着研究新东西，然后写写 教程啊，但是理想总是很美好的，但现实往往很残酷，我也要生活。 另外说到博客专家这个衔头，很多朋友喜欢拿我这个来黑我，其实并没有什么大用， 每个月原创超过10篇就能收到一本书而已，大部分是C币商城里的一些旧书...

5.小猪接下来的想做点什么？ 答：来一次说走就走的旅行，嗨一个月，然后等过年！ 好吧，我也想，可惜兜里没钱，接下来的日子嘛，想把公司的项目琢磨透，修下bug，然后学一些 其他的东西，接着写点小玩意玩玩，存钱买个机械键盘(ikbc G104)，复习下高数准备一月份 补考等等，然后过完年，可能跑深圳那边找找机会吧~可能偶尔会更新那么一两篇文章吧， 不过不要期望太大，进阶系列也不是说不写，只是暂时不会写，等找到一份稳定的工作，有了 一定经济能力，再开始写吧~

致谢：

嗯，好吧，唠唠叨叨地终于把自己肚子里的东西都吐出来了~



按照一般的套路，肯定要说一堆，谢谢ccav之类的吧，嗯！ 谢谢w3c鸟巢的站长FK对每篇文章的认真排版，以及小猪秘密基地里的基神，B神，街神，曹神等的 技术支持，还有一直默默支持小猪的各位朋友，在这里真挚的说一声感谢~ 好了，就说这么多吧，谨以此文纪念我将近两年的csdn博客生涯吧~

完结散花~ 是终点也是起点

to be continued... 待续

**</p